НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

(повне найменування інституту, факультету)

Автоматизованих систем обробки інформації і управління

(повна назва кафедри)

«До захисту допущено»
В.о. завідувача кафедри
_____ *Олександр ПАВЛОВ*
(підпис) (ініціали, прізвище)
“ ” 2020 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Програмне забезпечення інформаційних управляючих систем та технологій»

спеціальності «121 Інженерія програмного забезпечення»

на тему *Діалогова система інформування абітурієнтів*

Виконав: студент IV курсу, групи *ІП-63Бондар Вікторія Сергіївна* _____
(прізвище, ім’я, по батькові) (підпис)

Керівник *ас. Очеретяний О.К.* _____
посада,науковий ступінь,вчене звання,прізвище,і ім’я, по батькові (підпис)

Консультант з графічної документації *доц., к.т.н., Ліщук К.І.* _____
посада,науковий ступінь,вчене звання,прізвище,і ім’я, по батькові (підпис)

Рецензент:

доц., к.т.н., доц. Кисленко Ю.І.
посада,науковий ступінь,вчене звання,прізвище,ім’я, по батькові (підпис)

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2020 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет (інститут) *Інформатики та обчислювальної техніки*
(повна назва)

Кафедра *автоматизованих систем обробки інформації і управління*
(повна назва)

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – *121 Інженерія програмного забезпечення*

Освітньо-професійна програма – *Програмне забезпечення інформаційних управляючих систем та технологій*

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ *Олександр ПАВЛОВ*
(підпис)
“___” _____ 2020 р.

ЗАВДАННЯ
НА ДИПЛОМНИЙ ПРОЄКТ СТУДЕНТУ

Бондар Вікторії Сергіївні
(прізвище, ім’я, по батькові)

1. Тема проєкту «*Діалогова система інформування абітурієнтів*»

керівник проєкту *Очеретяний Олександр Констянтинович, ас.*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від “07” травня 2020 р. №1081-с

2. **Термін подання студентом проєкту** «08» *червня 2020 року*

3. Вихідні дані до проєкту

Технічне завдання

4. Зміст пояснювальної записки

1) Аналіз вимог до програмного забезпечення: основні визначення та терміни, опис предметного середовища, огляд існуючих технічних рішень та відомих програмних продуктів, розробка функціональних та нефункціональних вимог

2) Моделювання та конструювання програмного забезпечення: моделювання та аналіз програмного забезпечення, засоби розробки, технічні рішення, архітектура програмного забезпечення

3) Розгортання та впровадження програмного забезпечення

4) Керівництво користувача, методика випробувань програмного продукту

5. Перелік графічного матеріалу

1) Схема структурна класів програмного забезпечення

2) Схема структурна станів варіантів використань

3) Схема бази даних

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. **Дата видачі завдання** «10» *березня 2020 року*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	*Вивчення рекомендованої літератури*	*21.02.2020*	
2.	*Аналіз існуючих методів розв'язання задачі*	*03.03.2020*	
3.	*Постановка та формалізація задачі*	*12.03.2020*	
4.	*Аналіз вимог до програмного забезпечення*	*19.03.2020*	
5.	*Алгоритмізація задачі*	*23.03.2020*	
6.	*Моделювання програмного забезпечення*	*10.04.2020*	
7.	*Обґрунтування використовуваних технічних засобів*	*17.04.2020*	
8.	*Розробка архітектури програмного забезпечення*	*24.04.2020*	
9.	*Розробка програмного забезпечення*	*01.05.2020*	
10.	*Налагодження програми*	*08.05.2020*	
11.	*Виконання графічних документів*	*15.05.2020*	
12.	*Оформлення пояснювальної записки*	*22.05.2020*	
13.	*Подання ДП на попередній захист*	*28.05.2020*	
14.	*Подання ДП рецензенту*	*11.05.2020*	
15.	*Подання ДП на основний захист*	*08.06.2020*	

Студент _____ Вікторія БОНДАР
(підпис)

Керівник _____ Олександр ОЧЕРЕТЯНИЙ
(підпис)

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проєкт	2	
2	А4	КПІ.ІП-6302. 045430.02.81	Пояснювальна записка	51	
3	А4	КПІ.ІП-6302. 045430.03.91	Технічне завдання	10	
4	А4	КПІ.ІП-6302. 045430.04.13	Опис програми	10	
5	А4	КПІ.ІП-6202. 045430.05.51	Програма та методика тестування	6	
6	А3	КПІ.ІП-6202. 045430.06.34	Керівництво користувача	6	
7	А3	КПІ.ІП-6202. 045430.07.99.СС	Схема структурна варіантів використань	1	
8	А3	КПІ.ІП-6202. 045430.08.99.СБД	Схема бази даних	1	
9	А3	КПІ.ІП-6202. 045430.09.99.СС	Схема структурна класів програмного забезпечення	1	

Зм.	Арк.	ПІБ	Підп.	Дата	КПІ.ІП-6302.045430.00.90		
Розробн.		Бондар В.С.			Відомість дипломного проєкту	Літ.	Лист 1 / Листів 1
Керівн.		Очеретяний О.К					
Консульт						КПІ ім.Ігоря Сікорського кафедра АСОІУ гр. ІП-63	
Н/контр.		Ліщук К.І.					
В.о.зав.каф		Павлов О.А.					

АНОТАЦІЯ

Пояснювальна записка дипломного проекту складається з чотирьох розділів, містить 27 таблиць, 5 рисунки та 9 джерел – загалом 54 сторінки.

Об’єкт дослідження: нецільові діалогові системи з пошуковим підходом.

Мета дипломного проекту: розробити програмне забезпечення, що у вигляді Telegram чат-боту буде інформувати абітурієнтів

У першому розділі було проведено аналіз предметної області, визначено основні вимоги до програмного забезпечення, а також опис уже існуючих технічних рішень.

У другому розділі детально розглянуто всі складові розробленого застосунку, зроблено короткий опис класі в методів, наявних в розробці і оглянуто засоби безпеки результуючого програмного продукту.

У третьому розділі описано можливості тестування діалогових систем, складено детальний тестовий план і проведено тестування застосування згідно складеного тестового плану.

У четвертому розділі демонструється, як потрібно розгортати дане програмне забезпечення і де можна знайти детальний опис та інструкцію, як працювати із застосунком з боку користувача.

У додатках наведено: текст програмного коду.

КЛЮЧОВІ СЛОВА: ПРИРОДНЯ МОВА, ДІАЛОГОВА СИСТЕМА, ЧАТ-БОТ, АБІТУРІЄНТИ

ABSTRACT

The explanatory note of the diploma project consists of four sections, contains 21 tables, 5 figures and 9 sources - a total of 51 pages.

Object of research: non-target dialog systems with a search approach.

The purpose of the diploma project: to develop software that in the form of Telegram chatbot will inform applicants in natural language format.

In the first section the analysis of the subject area was carried out, the basic requirements to the software, and also the description of already existing technical decisions were defined.

The second section examines in detail all the components of the developed application, makes a brief description of the class in the methods available in the development and reviews the security tools of the resulting software product

In the third section the possibilities of testing of dialog systems are described, the detailed test plan is made and testing of application according to the made test plan is carried out.

The fourth section demonstrates how to deploy this software and where you can find a detailed description and instructions on how to work with the application by the user.

The appendices contain: the text of the program code.

KEY WORDS: NATURAL LANGUAGE, DIALOGUE SYSTEM, CHAT BOT, APPLICANTS

Пояснювальна записка
до дипломного проєкту

на тему: Діалогова система інформування абітурієнтів

Київ – 2020 року

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ 8

ВСТУП 9

1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 11

1.1 Загальні положення 11

1.2 Змістовний опис і аналіз предметної області 12

1.3 Аналіз успішних IT-проектів 13

1.3.1 Аналіз відомих технічних рішень 13

1.3.2 Аналіз відомих програмних продуктів 14

1.4 Аналіз вимог до програмного забезпечення 15

1.4.1 Розроблення функціональних вимог 16

1.4.2 Розроблення нефункціональних вимог 22

1.4.3 Постановка комплексу завдань модулю 23

1.5 Математичне забезпечення 24

1.6 Висновки по розділу 26

2 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 227

2.1 Моделювання та аналіз програмного забезпечення 227

2.2 Архітектура програмного забезпечення 29

2.3 Конструювання програмного забезпечення 31

2.4 Аналіз безпеки даних 34

2.5 Висновки по розділу 34

3 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 35

3.1 Аналіз якості ПЗ 36

3.2 Опис процесів тестування 439

3.3 Опис контрольного прикладу 48

4 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 49

4.1 Розгортання програмного забезпечення 49

4.2 Робота з програмним забезпеченням 49

ВИСНОВКИ **50**

ПЕРЕЛІК ПОСИЛАНЬ **51**

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

NLP (Natural language processing) – підгалузь лінгвістики, інформатики, інформаційної інженерії і штучного інтелекту, пов’язана із взаємодією комп’ютерів з людськими (природніми) мовами, в тому числі з тим, як програмувати комп’ютери для обробки і аналізу великих об’ємів даних природньої мови.

Токенізація – задача розбивання заданої послідовності символів чи певної одиниці документу на менші частини (токени), з можливим усуненням деяких символів, таких як пунктуація.

Стеммінг і лемматизація - націлені на зведення флективних, і деколи похідних форм слова до спільної базової форми, при чому лемматизація використовує словник і морфологічний аналіз.

Стоп-слова – це звичайні слова (токени), які не вносять великий вклад у зміст чи значення документу, тому при обробці тексту викидаються чи ігноруються.

Торба слів (bag-of-words) – модель спрощеного подання в обробці природніх мов, у якій текст (наприклад, речення або документ) представляється у вигляді торби (мультимножини) його слів, не беручи до уваги граматику і навіть порядок слів, але зберігаючи множинність.

ВСТУП

Саме зараз людство переживає свою інформаційну епоху розвитку. Щодня невловимо росте кількість інформації, яку ми передаємо і продукуємо. А з появою мережі Інтернет, зросла ще й швидкість передачі і доступність даних. Тобто інформації навколо стало так багато, що виникла гостра потреба у її систематизації та класифікації. Так як більшість ресурсів існує саме в текстовому форматі то набирають популярності різноманітні засоби аналізу тексту, в тому числі NLP.

З проблемами розширення інформаційного простору можна стикнутися у будь-якій сфері діяльності. Найближчим прикладом слугує наш університет. КПІ складається з понад двадцяти різних факультетів та інститутів, на яких навчається близько двадцяти тисяч студентів. Спираючись на все це, політех має одну із найбільш розвинутих систем комунікацій серед українських університетів. Тільки в мессенджері Telegram налічується тисячі публічних чатів та каналів, знайти серед яких релевантну інформацію дуже складно. Найбільше потерпають від цієї проблеми недосвідчені юзери «КПІшних інтернетів», а саме абітурієнти. Дуже легко заблукати серед такої великої кількості різнотипних джерел інформації.

Сьогодні, звичайно, існує дуже багато різних систем і способів інформування абітурієнтів. Проблема в них полягає в тому, що переважна більшість з них адмініструється живою людиною. Ті засоби, які автоматизовані, найчастіше мають дуже обмежений спектр інформації або працюють в рамках одного домену. Тому щоб досягнути високої швидкості передачі даних при взаємодії з користувачем і достатнього набору даних на обрану тематику, жоден такий сервіс не підходить.

Спираючись на подану інформацію, в рамках цього дипломного проекту було розроблено програмний комплекс для інформування абітурієнтів у вигляді Telegram чат-боту. Програма застосовує алгоритми семантичного аналізу

тексту і NLP, що дає змогу автоматизувати процес «спілкування» з користувачем і оперувати великою кількістю різнотипних даних.

Актуальність теми: постійне збільшення кількості існуючої інформації та необхідність в її агрегації та аналізі в зручному для користувача форматі.

Мета розробки: інформування абітурієнтів у форматі питання-відповідь, взаємодія з користувачем через зручний та відомий інтерфейс Telegram, надати можливість користувачу задавати питання у вільному форматі, виконувати аналіз будь-якої кількості різнотипних текстових даних, мінімізувати необхідність використовувати інші ресурси.

Завдання розробки: аналіз збережених в базі текстів, пошук найбільш підходящої відповіді, виведення результатів користувачу.

Практичне значення одержаних результатів: розроблений продукт реалізований у вигляді чат-боту Telegram і надає можливість отримувати повну і релевантну інформацію, мінімізуючи затрати на пошук і аналіз серед безлічі інших ресурсів, агрегуючи ці дані і дозволяє користувачу взаємодіяти з програмою у вільній формі повідомлень українською мовою.

1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Загальні положення

НТУУ «КПІ» ім. Сікорського щороку займає перші місця в рейтингу по кількості поданих заяв під час вступної кампанії. Тільки за 2019 рік на рахунок КПІ поступило понад тридцять шість тисяч заяв від абітурієнтів за різними напрямами навчання. Така популярність вимагає й високий рівень інформування. Це стосується не тільки процесу подання документів і вступної кампанії, а й самого навчання в університеті. Для майбутніх студентів цей етап є надзвичайно важливим, зазвичай від прийнятих рішень залежить щонайменше найближчі 4 роки життя і майбутній професійний розвиток. Тому питань є дуже багато і з-поміж великої кількості ресурсів знайти необхідну інформацію складно. В такому випадку допомагають студенти і волонтери, які можуть дати відповідь на поставлене питання або підкажуть, де її можна знайти. Тим не менш, людські ресурси вичерпні і рано чи пізно постає питання щодо автоматизації цього процесу.

Вище описана ситуація застосування так званого живого оператора, коли з користувачем працює людина. Проте в сучасному світі виникла тенденція замінювати людські ресурси комп’ютерними. Системи, які виконують роль автоматизованого оператора називаються діалоговими системами. Для робот таких систем раніше застосовувалися жорстко задані опції та команди, що погіршує сприйняття користувача. Тому більшість сучасних великих компаній застосовують розробки діалогових систем із застосуванням NLP. Обробка природньої мови дає змогу користувачу задавати питання програмі у вільній формі, що також збільшує критерії пошуку і гнучкість системи.

Тож ми отримали актуальну проблему і найбільш передовий спосіб вирішення проблем такого характеру. Тому ця дипломна робота є програмним рішенням, яке пов’язує в собі два цих випадки – діалогова система, яка інформує абітурієнтів і застосовує NLP.

1.2 Змістовний опис і аналіз предметної області

Діалогові системи прийнято поділяти на цільові (goal-oriented) та нецільові (non-goal-oriented).

Цільові діалогові системи вирішують конкретну задачу користувача і, як правило, взаємодіють із зовнішнім сховищем даних. Такі системи зазвичай складаються з наступних компонентів:

- аналізатор природньої мови (natural language understanding) змінює висловлювання користувача в машинно-читаєму структуру даних;
- трекер стану діалогу (dialogue state tracker) визначає теперішній стан діалогу, який використовується для вибору наступної дії;
- система визначення дії (dialogue policy learning) обирає наступну дію діалогової системи в залежності від теперішнього стану;
- генератор природньої мови (natural language generator) змінює результат дії з машинно-чатаємої структури в людську мову.

Кожна цільова діалогова система створюється і налаштовується від конкретну задачу. При цьому часто деякі або всі компоненти створюються інженерним методом, з більшим числом вручну створених правил. Такі системи неможливо переналаштувати під інші задачі. У теперішній час створення універсальної діалогової системи є відкритою проблемою.

Нецільові діалогові системи, які ще називають чат-ботами, не намагаються вирішити конкретну задачу. Їх ціль в тому, щоб підтримати розмову з користувачем на визначену або вільну тему. Існує два основних підходи для побудови нецільових діалогових систем:

- генеративний – відповідь на повідомлення користувача породжується за допомогою деякої моделі. В теперішній час найбільш популярним способом побудови таких систем - seq2seq-моделі;
- пошуковий – відповідь на повідомлення користувача вибирається із великого набору готових відповідей. Як правило, для повідомлення і всіх

відповідей будуються векторні представлення однієї розмірності, після чого відповідь обирається у відповідності з деякою метрикою.

1.3 Аналіз успішних ІТ-проектів

1.3.1 Аналіз відомих технічних рішень

Щоб діалогова система була готовою до використання на приктиці, вона повинна задовільняти наступні вимоги:

- швидкий час відповіді. Користувач очікує від діалогової системи тієї ж швидкості реакції, що і від звичайної людини;
- надійність. Користувач спілкується з діалоговою системою у зручний для нього час. Якщо колись діалогова система не буде спроможна дати відповідь через перезавантаження чи профілактичні роботи, користувач може ніколи більше до неї не звернутися;
- легкість маштабування. Діалогова система за короткий проміжок часу може стати доволі популярною, при цьому також можливі значні скорочення навантаження, наприклад, у нічний час. Необхідно ефективно використовувати обчислювальні ресурси, щоб не допускати ні сповільнення роботи під навантаженням, ні простою великої кількості потужностей.

Для досягнення вище описаних вимог і при цьому доступності усіх необхідних інструментів для аналізу даних, в тому числі текстових, зазвичай використовують мову програмування Python.

Python - інтерпретована мова програмування, яка має неймовірно велику кількість потужних засобів і пакетів для аналізу даних та математичних обчислень. При роботі з NLP використовують такі інструменти:

- pandas для різного роду маніпуляцій з даними (у вигляді таблиць або числових рядів) та їх аналізу;
- nltk (Natural Language Toolkit) – провідна платформа для створення NLP-програм на Python, в якій є легкі у використанні інтерфейси для багатьох

мовних корпусів, а також бібліотеки для обробки текстів для класифікації, токенізації, стеммінгу, розмітки, фільтрації семантичних суджень;

- pymorphy2 – морфологічний аналізатор та генератор для російської та української мов;
- sklearn.feature_extraction.text - підмодуль, в якому містяться утиліти для побудови частотних векторів з текстових документів.

Останнім часом багато популярних веб-сервісів надають стороннім розробникам програмні інтерфейси для розробки діалогових системЮ інтегрованих у цей сервіс. В тому числі такі інтерфейси надають такі соціальні мережі і мессенджери, як Facebook, WhatsApp, Telegram, Viber та інші. Найбільш популярним в цьому плані на сьогодні є Telegram. Для роботи з Telegram Bot API використовується бібліотека pytelegrambotapi.

1.3.2 Аналіз відомих програмних продуктів

На сьогодні діалогові системи користуються великою популярністю. До таких систем з використанням NLP можна віднести й сервіси з розпізнаванням мовлення. Найпопулярніші з них – це, наприклад віртуальні помічники Cortana (Windows), Siri (Apple), Alexa (Amazon), Alisa (Yandex).

Якщо говорити саме про чат-боти, то найпершим розробленим чат ботом була діалогова система-психоаналітик ELIZA. Приблизно в ті ж роки винайшли віртуального співбесідника PARRY для вивчення шизофренії. Ці, і схожі до них системи, є Closed Domain, тобто здатні говорити тільки на одну чітко визначену тему.

Більш сучасним прикладом діалогової системи є Xiaoice - віртуальний співбесідник з широким використанням систем штучного інтелекту. За допомогою складних обчислювальних алгоритмів, використовуючих хмарні обчислення та Big Data, розвинувся в складну систему реалізації емоційного інтелекту (EQ).

Ще одним прикладом такої системи є A.L.I.C.E. – віртуальний співбесідник, створена по подобі до ELIZA але з використанням техніки евристичного співставлення фрази користувача зі зразками в базі знань. Один із найкращих існуючих діалогових систем і не має жорсткої прив'язки до теми, заежить тільки від бази знань.

Аналоги діалогових систем у вигляді чат-ботів Telegram:

- Fitmeal chatbot – допомагає слідкувати за харчуванням, рахує калорії, видає статистику. Як чатбот, використовує підхід пошуку по ключовим словам, тому функціонал є обмеженим;
- H&M chat bot – підбирає одяг з однойменного бренду. Реагує лише на певні чітко визначені слова і команди;
- BookFlights chatbot - бот для бронювання і покупку квитків на літак. Також використовує командний підхід і є обмеженим;
- HealthTap chatbot – надає консультацію по питанням з приводу стану здоров'я. Використовує підхід пошуку по ключових словах заданого питання серед уже існуючих в базі відповідей лікарів. Надає широку варіативність можливих відповідей, серед яких користувач може знайти підходящу. Підхід є дуже неточним.

Дослідивши область розробки, можна прийти до висновку, що не існує жодної системи, яка б одночасно виконувала роль автоматизованого діалогового чат-боту і відповідала заданій тематиці – інформування абітурієнтів. Переважна більшість будь-яких інформаційних ресурсів, пов'язаних зі вступом і навчанням у КПІ, адмініструються людиною, або автоматично оновлюють дані з певного конкретного домену, при цьому інформація подається не структуризовано.

1.4 Аналіз вимог до програмного забезпечення

Система повинна підтримувати такі системні ролі, як користувач і адміністратор.

Користувач має доступ до функціоналу сервісу: задавати боту конкретні визначені команди, задавання питань природньою мовою, отримання результатів відповіді, самостійнр вносити посилання для поповнення датасету.

Виокремимо дві основні лінії з яких складається програмне забезпечення:

- взаємодія з програмним інтерфейсом Telegram;
- метод, який приймає на вхід ідентифікатор користувача і повідомлення, повертаючи відповідь.

1.4.1 Розроблення функціональних вимог

Схема структурна варіантів використання наведена у графічному матеріалі. В системі передбачено наступні варіанти використання, описані у таблицях:

Таблиця 1.1 - Варіант використання UC001

Назва	Ведення документів з інформацією
Опис	Адміністратор може додавати і видаляти текствові документи з інформацією вручну
Учасники	Адміністратор

Таблиця 1.2 - Варіант використання UC002

Назва	Додавання посилань на джерела інформації адміністратором
Опис	Адміністратор за допомогою спеціальної команди вносить в бот веб-посилання на відповідне джерело інформації
Учасники	Адміністратор
Передумови	Виникла необхідність поповнити датасет новими актуальними даними

Продовження таблиці 1.2

Постумови	В базу даних додано посилання, а інформація з відповідної цьому посиланню веб-сторінки конвертується у текстовий документ і поповнює датасет з аналізованою інформацією
Основний сценарій	Адміністратор виконує команду /add_link і вносить необхідне посилання, як повідомлення

Таблиця 1.3 - Варіант використання UC003

Назва	Видалення неактуальної інформації з датасету вручну
Опис	Адміністратор за допомогою завчасно підготованих скриптів або вручну з текствих фалів видаляє інформацію, яку вважає уже неактуальною або неправильною
Учасники	Адміністратор
Передумови	Деяка інформація в текстових джерелах стала неактуальною або невірною
Постумови	Модифікація або видалення текстового файлу в датасеті, на роботу алгоритму не впливає
Основний сценарій	Адміністратор запускає скрипт, або відкриває необхідний текстовий файл і модифікує дані в ньому, або просто видаляє відповідний файл

Таблиця 1.4 - Варіант використання UC004

Назва	Початок роботи з ботом
Опис	Користувач запускає бот і починає роботу з ним
Учасники	Користувач

Продовження таблиці 1.4

Передумови	Виникла необхідність скористатися сервісом
Постумови	Бот запущений іготовий до роботи з цим конкретним користувачем
Основний сценарій	Користувач зайшов у інтерфейс боту і натиснув команду «start»

Таблиця 1.5 - Варіант використання UC005

Назва	Задавання питання в чат-бот
Опис	Користувач може задати цікавляче питання у вільній формі
Учасники	Користувач
Передумови	Користувач обрав команду «Задати питання»
Постумови	Бот було запущено
Основний сценарій	Користувач пише повідомлення у текстовому вікні чат-боту

Таблиця 1.6 - Варіант використання UC006

Назва	Перегляд списку питань, які найчастіше задаються
Опис	Користувач може переглянути список питань, які найчастіше задавалися боту
Учасники	Користувач
Передумови	Користувач обрав команду «Подивитися найчастіше задавані питання»
Постумови	Виведення списку найбільш популярних команд
Основний сценарій	Користувач обирає команду «Подивитися найчастіше задавані питання»

Таблиця 1.7 - Варіант використання UC007

Назва	Внесення оцінки відповідності результатів
Опис	Користувач може за бажанням поставити оцінку того, наскільки отримана відповідь задовільняє умови пошуку
Учасники	Користувач
Передумови	Користувач обрав команду «Поставити оццінку»
Постумови	Поставлена оцінка системи з точки зору користувача
Основний сценарій	Користувач обирає команду «Поставити оцінку» і натискає на одну з відповідних оцінок у цифрах 1-5

Таблиця 1.8 - Варіант використання UC008

Назва	Додавання посилань на джерела інформації користувачем
Опис	Користувач за допомогою спеціальної команди вносить в бот веб-посилання на відповідне джерело інформації
Учасники	Користувач
Передумови	Користувач знайшов підходяще по тематиці джерело у вигляді веб-сторінки і захотів внести його до бази знань
Постумови	В базу даних додано посилання, а інформація з відповідної цьому посиланню веб-сторінки конвертується у текстовий документ і поповнює датасет з аналізованою інформацією
Основний сценарій	Користувач виконує команду /add_link і вносить необхідне посилання, як повідомлення

Нижче приведено таблиці, в яких описані функціональні вимоги:

Таблиця 1.9 - Опис функціональної вимоги REQ001

Номер	REQ001
Назва	Веденя джерел інформації
Опис	Адміністратор може додавати і видаляти джерела інформації вручну

Таблиця 1.10 - Опис функціональної вимоги REQ002

Номер	REQ002
Назва	Наявність команди для початку роботи з ботом
Опис	Користувач повинен мати змогу почати роботу з ботом за допомогою відповідної команди.

Таблиця 1.11 - Опис функціональної вимоги REQ003

Номер	REQ003
Назва	Приймати на вхід повідомлення для подальшого опрацювання
Опис	Кристувач пише повідомлення в бот, як до звичайного співрозмовника, далі це повідомлення передається на вхід у функцію, яка виконує аналіз

Таблиця 1.12 - Опис функціональної вимоги REQ004

Номер	REQ004
Назва	Виведення списку начстіше задаваних питань
Опис	При виборі відповідної команди користувачем, програма виводить список найчастіше задаваних питань

Таблиця 1.13 - Опис функціональної вимоги REQ005

Номер	REQ005
Назва	Виставлення оцінок коректності роботи системи
Опис	Користувач може виставити оцінку того, наскільки він задоволений знайденою відповіддю

Таблиця 1.14 - Опис функціональної вимоги REQ006

Номер	REQ006
Назва	Внесення нових джерел інформації у вигляді веб-посилань
Опис	Користувач або адміністратор може при бажанні чи необхідності додавати лінки на джерела інформації

Таблиця 1.15 - Опис функціональної вимоги REQ007

Номер	REQ007
Назва	Видалення нерелевантної інформації
Опис	Користувач або адміністратор може при бажанні чи необхідності додавати лінки на джерела інформації

Взаємозв’язки між вимогами клієнтського застосунку відображені на рисунку 1.1.

Рисунок 1.1 - Матриця залежності між вимогами застосунку і варіантами використання

1.4.2 Розроблення нефункціональних вимог

Вимоги до інтерфейсу:

- інтерфейс побудовано на базі Telegram Bot API;
- наявні команди для роботи базових функцій програми.

Апаратні та програмні вимоги:

- наявність інтернету та необхідної операційної системи або браузера для відкриття мессенджеру Telegram;

1.4.3 Постановка комплексу завдань модулю

Основна ціль розробки – інформування абітурієнтів у вигляді діалогової системи із застосуванням Telegram API, представляючи собою систему, яка агрегує в собі велику кількість різнотипних даних за заданою тематикою.

Для досягнення мети даної роботи, система повинна вирішувати наступні задачі:

- взаємодія з користувачем через API мессенджеру Telegram;
- виконання найпростіших команд, таких як запуск боту;
- розпізнавання та аналіз заданого питання у повідомленні;
- семантичний аналіз збережених текстових файлів;
- пошук релевантної відповіді на поставлене запитання;
- видача результатів пошуку користувачу;
- аналіз і збереження найбільш часто задаваних питань;
- внесення веб-посилання на підходяще джерело інформації.

1.5 Математичне забезпечення

В даній роботі основою пошуку відповідей лежить алгоритм TF-IDF, тому важливо розглянути, що саму він собою уявляє.

TF-IDF (term frequency - inverse document frequency) — статистичний показник, що використовується для оцінки важливості слів у контексті документа, що є частиною колекції документів чи корпусу. Вага (значимість) слова пропорційна кількості вживань цього слова у документі, і обернено пропорційна частоті вживання слова у інших документах колекції.

Основні переваги цієї метрики:

- слова, які не важливі для будь-якого документу, наприклад прислівники чи вигуки – отримають дуже низьку вагу TF-IDF (тому що часто зустрічаються абсолютно у всіх документах), а важливі – високу;
- нею доволі просто користуватися і не вимагає громістких серйозних обчислень.

Застосовувати її варто для виявлення важливих слів і стоп-слів в документах. Деякі розширення цієї формули можна використовувати для покращення роботи класифікаторів тональності.

TF – term frequency – частотність терміну, яка вимірює, наскільки часто той чи інший термін зустрічається в документі. Чим більший документ – тим більша ймовірність, що конкретне слово може зустрічатися частіше, тому використовуються відносні обрахунки. В даному випадку кількість разів, коли потрібний термін зустрівся в тексті ділиться на загальну кількість слів у тексті.

$$\mathrm{TF} = \frac{n_i}{\sum_k n_k}, \tag{1.1}$$

де n_i є число входжень слова в документ, а в знаменнику - загальна кількість слів в документі.

IDF - inverse document frequency – обернена частотнысть документів. Вона вимірює безпосередньо важливість терміну. Тобто якщо при розрахунках TF всі терміни вважалися ріними по важливості, то в реальних даних прийменники, наприклад, зустрічаються дуже часто, але на практичний сенс тексту майже не впливає. Для вирішення цієї проблеми і існує IDF. Він рахується, як логарифм від загальної кількості документів, поділеного на кількість документів, в яких зустрічається певний конкретний термін.

$$\mathrm{IDF} = \log \frac{|D|}{|(d_i \supset t_i)|}, \tag{1.2}$$

де |D| - кількість документів колекції, а $|(d_i \supset t_i)|$ - кількість документів, в яких зустрічається слово t_i (коли $n_i \neq 0$).

Міра TF-IDF часто використовується для подання документів колекції у вигляді числових векторів, що відображають важливість використання кожного слова з деякого набору слів (кількість слів набору визначає розмірність вектора) в кожному документі. Подібна модель називається векторною моделлю і дає можливість порівнювати тексти, порівнюючи їх представляють вектора в певний метриці (евклідова відстань, косинусна міра, манхеттенська відстань, відстань Чебишова та інші), тобто виконувати кластерний аналіз.

1.6 Висновки по розділу

В рамках першого розділу було розглянуто особливості предметної області, опис сучасних методів вирішення поставленої проблематики. Розглянуто характеристики і недоліки найближчих по своїй суті продуктів. Сформульовано функціональні і нефункціональні вимоги до програмного забезпечення.

2 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Моделювання та аналіз програмного забезпечення

Розроблене програмне забезпечення складається з двох окремих основних частин. Перша частина відповідає за взаємодію з програмним інтерфейсом Telegram. В ній відбувається вся взаємодія з користувачем. Кожне повідомлення, отримане від користувача відправляється на обробку в другу частину. Друга частина відповідає за обробку питання, аналіз текстових документів та пошук відповіді. Відповідний метод приймає на вхід ідентифікатор користувача та повідомлення з API, і повертає необхідну відповідь.

Схема питання-відповідь зазвичай складається з трьох процесів – обробка питання, інформаційний пошук і обробка відповіді. Ці кроки детально демонструє рисунок 2.1.:

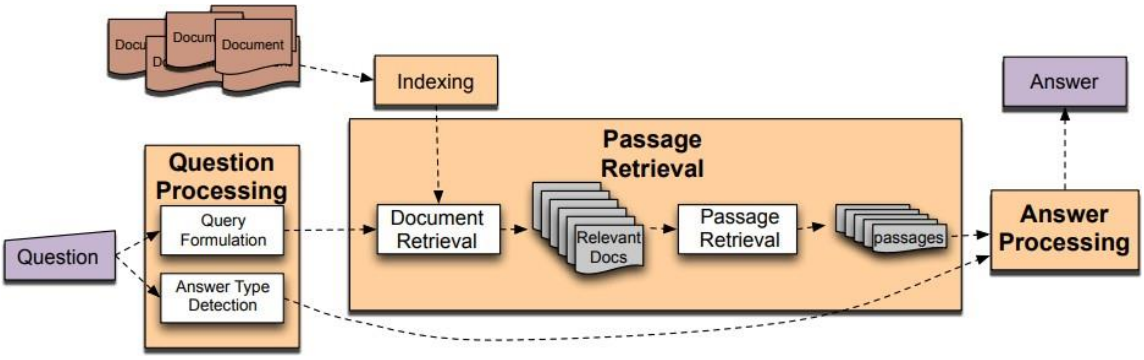


Рисунок 2.1 – Схема функціонування діалогової системи

Ключовим процесом розробки є аналіз та обробка текстів за допомогою NLP. Розглянемо ключові кроки названого процесу:

- токенізація по реченням – процес розділення письмової мови на речення-компоненти;

- токенізація по словах – процес розділення речення на слова-компоненти;
- Лемматизація та стеммінг тексту – приведення всіх зустрічних слоформ до однієї, нормальної словарної форми;
- виокремлення стоп-слів – викидування з тексту слів, які можуть слугувати шумом і заважати семантичному аналізу;
- виокремлення зайвих знаків пунктуації за допомогою регулярних виразів;
- застосування методу TD-IDF для генерації відповіді – це статистична міра для оцінки вадливості слова в документі, який є частиною колекції чи корпусу.

Також наявний фонкціонал, який надає можливість внесення веб-посилань на можливі джерела інформації через інтерфейс боту. За допомогою спеціальної команди, користувач або адміністратор може ввести веб-посилання, яке далі програмно парситься у текстовий файл і заноситься до базит даних. Надалі цей фал може брати участь у аналізі та та веведенні відповідей.

На рисунку 2.2 зображено модель та нотації бізнес-процесів системи.

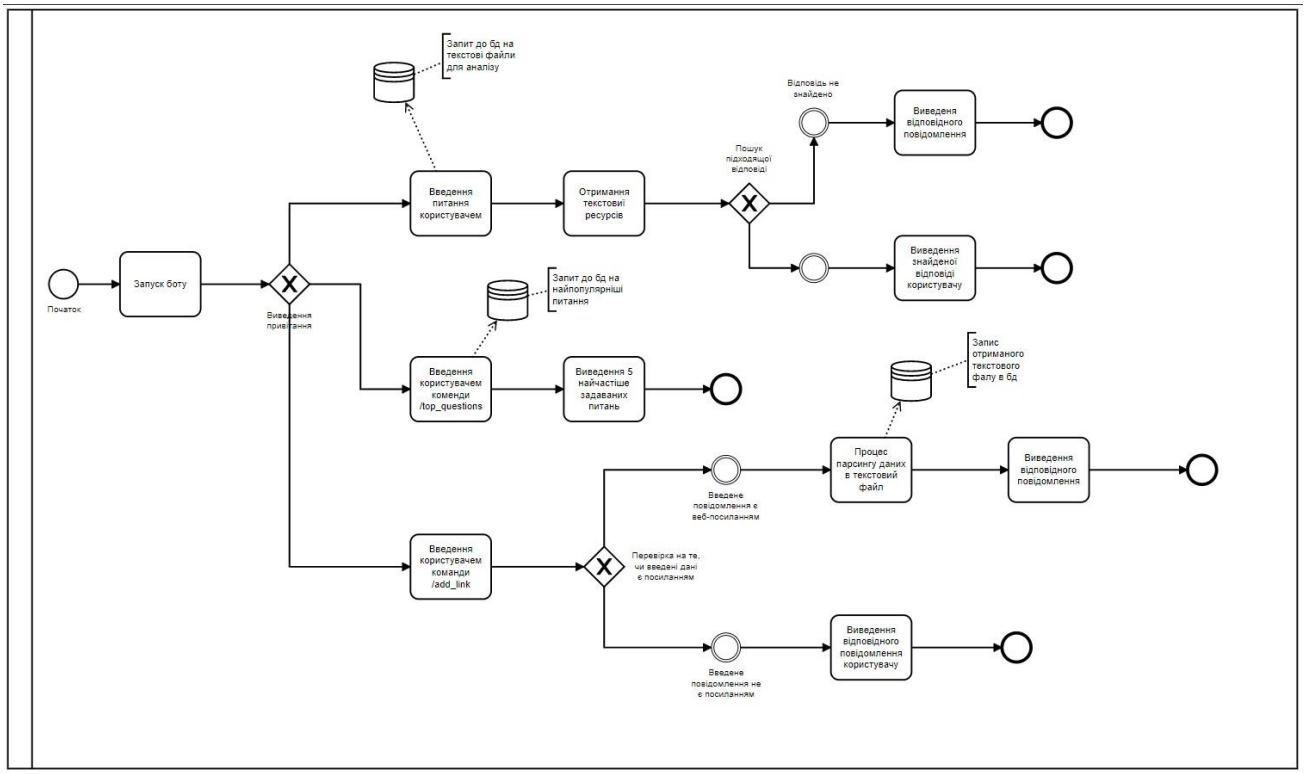


Рисунок 2.2 –Схема бізнес-процесів систем

2.2 Архітектура програмного забезпечення

Як було сказано вище, це програмне забезпечення ділиться на дві частини - інтерфейс взаємодія з Telegram API і серверна частина для обробки тексту. Обидві частини написані на мові програмування Python із застосовуванням як основного інструменту NLKT платформи для роботи з обробкою природньої мови.

Програмний комплекс призначений для виконання на операційній системі Windows. Робота з інтерфейсом Telegram відбувалася за допомогою бібліотеки pyTelegramBotAPI.

Система є класичним представником классу three-tier architecture, тобто розроблена як трьох-ланцюгова система орієнтована на клієнт-серверну взаємодію.

Отже система поділяється на 3 компоненти:

– клієнтські розширення для застосунку Telegram, так звані - боти, які дають змогу користувачам цього сервісу взаємодіяти із зовнішніми сервісами;

– серверний застосунок, який відповідає за обробку запитів користувачів, формування відповідей на основі інформації в базі даних та описаних в роботі алгоритмів;

– база даних, яка необхідна для роботи із даними застосунку, в тому числі використовується для збереження результатів роботи програми для подальшої статистичної обробки, якщо така обробка буде необхідною.

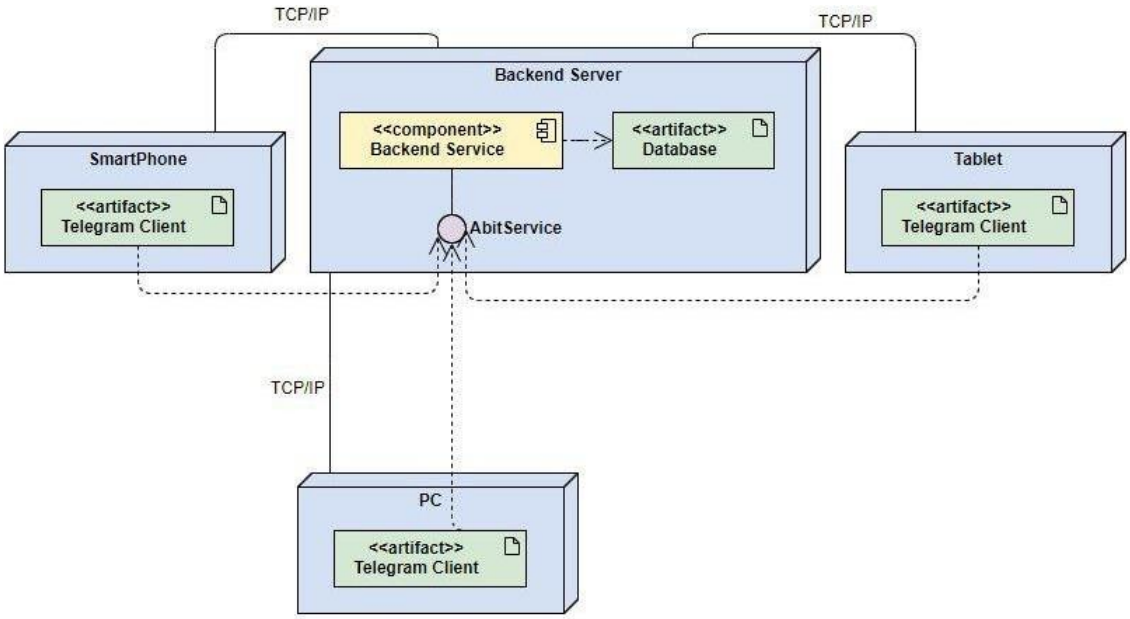


Рисунок 2.2 – Схема структурна компонентів

Проект системи створено з урахуванням можливого подальшого горизонтального та вертикального масштабувань, в тому числі і перенесення серверної частини та бази даних до хмарного сховища.

Взаємодію користувача із системою в компонентному розрізі можна показати на наступній послідовній діаграмі (sequence diagram):

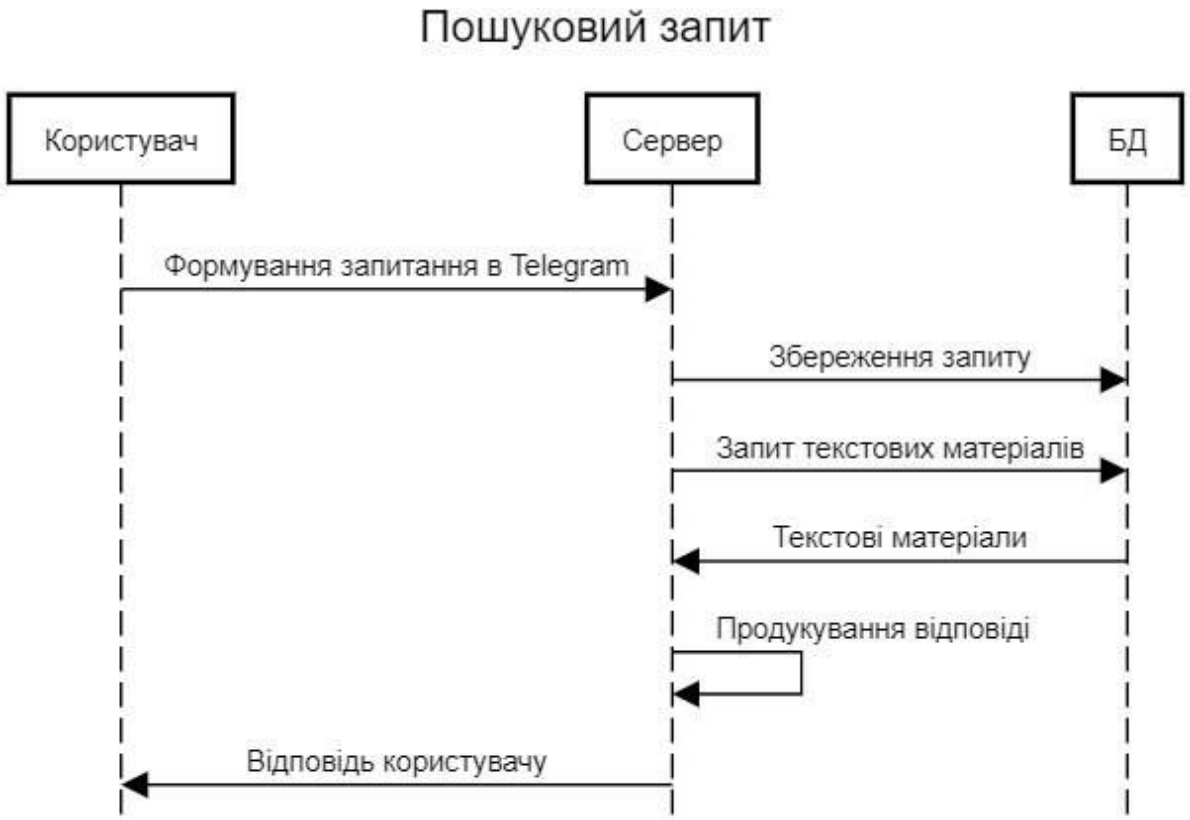


Рисунок 2.3 – Схема структурна послідовності пошукового запиту

Детальніше розглянемо цей сценарій, та його візуальну інтепретацію:

- користувач формує запит в Telegram;
- сервер зберігає запит до бази даних;
- відбувається аналіз вхідного запиту;
- запит текстових матеріалів з бази;
- отримання текстових матеріалів;
- аналіз тексту і пошук підходящої відповіді;
- надсилання відповіді користувачу.

2.3 Конструювання програмного забезпечення

Діаграма класів програмного продукту наведена у графічному матеріалі. Детальний опис класів та функцій наведений нижче в таблицях.

Таблиця 2.1 - Опис класів сервісу

Клас	Опис
DocumentItem	Клас-модель бази даних для зберігання документів
LinkItem	Клас-модель бази даних для зберігання ресурсних веб-посилань
IncomingQuestion	Клас-модель бази даних для зберігання вхідних запитань
UserScore	Клас-модель бази даних для збереження оцінок користувача
DBDataSaver	Надає функціонал для збереження даних в базу данихя
DataProcessor	Виконує попередню обробку тексту – токенізацію, лемматизацію і тд
TelegramConnector	Здійснює підключення до Telegram API
BotAPI	Виконує роботу з користувачем та передає питання і відповіді до серверної частини
SourceParser	Клас, який реалізовує зчитування веб-посилання, як можливого ресурсу даних і запис отриманої інформації у текстовий файл

Таблиця 2.2 - Опис методів класів сервісу

Клас	Метод	Опис
TelegramConnector	create_connection	Створює з’єдання з Telegram API
TelegramConnector	check_connection	Відповідає за перевірку з’єднання з Telegram API

Продовження таблиці 2.2

DataProcessor	summarize	Виконує суммаризаці тексту за допомогою процесів токенізації по реченням і по словам, лемматизації та очистки від стоп-слів
DataProcessor	find_answer	Використовуючи модель bag-of-words та метод TD-IDF знаходимо найбільш підходящу відповідь
DataProcessor	clean_text_round	За допомогою регулярних виразів чистить текст від непотрібних символів
BotAPI	start_message	Виконує запуск боту користувачем, початок роботи
BotAPI	sent_greetings	Розпізнає привітання і відповідає на них
BotAPI	get_question	Виконує зчитування питання від користувача та передає на аналіз
BotAPI	set_response	Отримує відповідь на питання та виводить її користувачу
BotAPI	get_score	Реалізовує функціонал оцінювання роботи боту користувачем, записує оцінку в базу даних
BotAPI	get_source_link	Отримує від користувача веб-посилання на можливий ресурс даних і відправляє і відправляє в парсер
SourceParser	simple_get	Надсилає HTTP Get запит до отриманого посилання і повертає відповідь на нього

Продовження таблиці 2.2

SourceParser	is_good_response	Перевіряє чи запит вернув HTML формат відповіді
SourceParser	log_error	Веде облік помилок, які виникли під час виконання програми
SourceParser	load_data	Записує отримані дані з HTML розмітки у текстовий файл
SourceParser	save_file	Зберігає записаний файл в базу даних

Важливу роль в системі відіграє база даних. Наглядна схема компонентів бази даних та зв’язки між ними наведена у документі КПІ.ІП-6302.045430.08.99.СБД “Схема бази даних”. А нижче наведено ряд таблиць, які детальніше описують таблиці бази та їх поля.

Таблиця 2.3 - Опис таблиці User

Колонка	Опис	Тип даних	Ключ
User ID	Ідентифікатор користувача	Integer	PK
Login	Ім’я користувача	Varchar(50)	
Mark	Оцінка, яку користувач поставив системі	Integer	

Таблиця 2.4 - Опис таблиці Message

Колонка	Опис	Тип даних	Ключ
Message ID	Ідентифікатор повідомлення	Integer	PK

Продовження таблиці 2.4

User ID	Посилання на користувача	Integer	FK
Text	Сам текст повідомлення	Varchar(1024)	
Date	Дата коли повідомлення було надіслано	timestamp	

Таблиця 2.5 - Опис таблиці Question

Колонка	Опис	Тип даних	Ключ
Question ID	Ідентифікатор запитання	Integer	PK
Message ID	Посилання на повідомлення, в якому запитання було задане	Integer	FK

Таблиця 2.6 - Опис таблиці Answer

Колонка	Опис	Тип даних	Ключ
Answer ID	Ідентифікатор запитання	Integer	PK
Message ID	Посилання на повідомлення, до якого буде надаватися відповідь	Integer	FK
TextEntity ID	Посилання на текстовий документ, в якому цю відповідь було знайдено	Integer	FK

Таблиця 2.7 - Опис таблиці TextEntity

Колонка	Опис	Тип даних	Ключ
TextEntity ID	Ідентифікатор документу	Integer	PK
Name	Назва документу	Varchar(250)	
Path	Повний шлях до документу	Varchar(250)	
Text	Текст для обробки	Varchar(1024)	
Questions ID	Посилання на запитання, на які були знайдені відповіді у конкретному документі	Integer[]	FK

Таблиця 2.8 - Опис таблиці History

Колонка	Опис	Тип даних	Ключ
Question ID	Ідентифікатор запитання	Integer	PK, FK
Answer ID	Ідентифікатор відповіді	Integer	PK, FK
User ID	Ідентифікатор користувача	Integer	PK, FK

2.4 Аналіз безпеки даних

Дані про повідоблення і дані мережі Telegram збергіються в базу даних застосунку через API-інтерфейси й вже готові оболонки для роботи з ним. А

також за допомогою веб-скрапінгу, що також реалізується з допомогою сторонніх бібліотек. Основою гарантії безпеки даних є з'єднання по надійних мережевих протоколах.

2.5 Висновки по розділу

В розділі 2 виконано моделювання та аналіз розробленого продукту, описано його архітектуру (специфікацію методів класів програмного забезпечення та функцій модулів) в діаграмі класів.

3 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Аналіз якості ПЗ

Тестування – це процес оцінки системи або її компонентів з ціллю визначити чи задовільняє вона вказаним вимогам.

Метою правильного тестування є не виявлення багів в коді чи функціоналі, а зниження ризику шляхом активного пошуку та усунення проблем, які могли б найбільше вплинути на користувача , який використовує дане програмне забезпечення.

Базові ідеї процесу тестування ПЗ:

- пріоритизація частин програмного продукту, які мають найбільший вплив на користувача;
- проведення набору тестів для перевірки необхідної функціональності;
- логування результатів;
- пріоритизація дефектів, базована по критичності;
- виправлення дефектів з найвищою критичністю;
- збирання дефектів з найменшим впливом, щоб вирішити пізніше.

Кожна специфічна система потребує свій окремий підхід до тестування. Так само й різні діалогові системи тестуються по по-різному.

Цільові діалогові системи можна оцінити як по відгуках користувачів, так і по критеріям, які відповідають завершенню цілі – наприклад чи вийшло у користувача отримати бажану відповідь на питання і з якої спроби йому це вдалося.

Нецільові діалогові системи можна оцінювати по метрикам перекриття, які зазвичай використовуються для автоматичного перекладу і суммаризації, такими як BLEU і ROUGE. Щоправда, було показано, що ці метрики практично

не корелюються з оцінками користувачів, як, таким чином, залишаються єдиним розумним способом оцінки таких систем.

Щоби діалогова система була готова до використання на практиці, вона повинна задовольняти вимогам наведеним нижче.

- Швидкий час відповіді. Користувач очікує від діалогової системи тієї ж швидкості реакції, що й від звичайної людини. Тому обробляти запит протягом хвилини або декількох хвилин просто недопустимо.
- Надійність. Користувач спілкується з діалоговою системою у зручний для нього час. Якщо колись діалогова система не відповість через перевантаження чи профілактичні роботи, користувач може більше ніколи до неї не звернутися.
- Легкість масштабування. Діалогова система за короткий проміжок часу може швидко стати популярною, при цьому також можливі значні зниження навантаження, наприклад, у нічний час. Необхідно ефективно використовувати обчислювальні ресурси, щоби не допускати ні сповільнення роботи від навантаженням, ні простою великої кількості потужностей.

3.2 Опис процесів тестування

Для проведення тестування, було розроблено ряд тест-кейсів. Умовно їх можна поділити на загальні, функціональні та спеціальні. Тестування відбувалося у десктопній версії Telegram з хорошим з'єднанням з мережею Інтернет.

3.2.1 Тестування загальних тест-кейсів

Таблиця 3.1 - Загальні тест-кейси

Ідентифікатор	Опис
GC-1	Перевірити процес запуску боту
GC-2	Перевірити можливість обрати команду
GC-3	Перевірити чи надсилаються повідомлення

Продовження таблиці 3.1

Ідентифікатор	Опис
GC-4	Перевірити можливість отримувати повідомлення у відповідь
GC-5	Перевірити можливість зупинити бот при різних виконуваних операціях
GC-6	Перевірити можливість видалення історії повідомлень з ботом у робочому режимі та після його зупинення
GC-7	Перевірити роботу боту, коли застосування знаходиться у фоновому режимі

3.2.2 Тестування функціональних тест-кейсів

Таблиця 3.2 - Функціональні тест-кейси

Ідентифікатор	Опис
FC-1	Перевірити, що бот розпізнає привітання і вітається у відповідь
FC-2	Перевірити можливість вибору команди /add_link
FC-3	Перевірити додавання веб-посилання у базу для подальшого аналізу
FC-4	Перевірити, що посилання, яке додається, розпізнається програмою
FC-5	Перевірити, чи файл записався не порожнім
FC-6	Перевірити можливість вибору команди /top_questions
FC-7	Перевірити результати команди /top_questions
FC-8	Перевірити чи правильно обраховуються найбільш часто задавані питання
FC-9	Перевірити можливість вибору команди /rate
FC-10	Перевірити можливість поставити оцінку наскільки надана відповідь розкриває задане питання

Продовження таблиці 3.2

Ідентифікатор	Опис
FC-11	Перевірити отримання відповіді на задане користувачем питання
FC-12	Перевірити чи надана відповідь була знайдена в текстовій базі даних і виведена коректно
FC-13	Перевірити чи буде відрізнятись відповіді, при багаторазовому введенні одного і того ж запитання
FC-14	Перевірити збереження та запис питань користувача до внутрішньої бази даних
FC-15	Перевірити збереження та запис ресурсних веб-посилань до внутрішньої бази даних
FC-16	Перевірити коректність парсингу інформації по веб посиланню у текстовий файл
FC-17	Перевірити збереження та запис отриманого в результаті парсингу текстового до внутрішньої бази даних
FC-18	Перевірити чи проводиться аналіз у новостворених текстових файлах в результаті роботи з ботом
FC-19	Перевірити збереження та запис оцінок користувача до внутрішньої бази даних

3.2.3 Тестування спеціальних тест-кейсів

До цього тест-кейси були розраховані на тестування функціональності програмного продукту. Спеціальні тест-кейси в даній роботі являють собою етапи тестування роботи саме алгоритмів і розрахунків. Для усіх алгоритмічних операцій ми використовуємо спеціальні бібліотеки. Тому дуже важливо перевірити коректність виконання зазначених там алгоритмів саме для цього програмного продукту і наших даних.

Таблиця 3.3 - Спеціальні тест-кейси

Ідентифікатор	Опис
SC-1	Перевірка коректності токенізації по реченням на конкретному тестовому документі
SC-2	Перевірка коректності токенізації по словам для конкретного тестового прикладу
SC-3	Перевірка коректності лемматизації для конкретного тестового прикладу
SC-4	Перевірка процесу пошуку і видалення стоп-слів з конкретного тестового прикладу
SC-5	Перевірка пошуку та видалення зайвих символів за допомогою регулярних виразів для конкретного тестового прикладу
SC-6	Перевірити коректність обчислень для методу TF-IDF шляхом порівняння обчислень, зроблених мануально вручну, з результатом, який нам видає бібліотечний метод TF-IDF для двох завчасно підготовлених тестових прикладів
SC-7	Серед двох створених тестових прикладів, обрати в файлах частину, яка слугувала б найбільш підходящою відповіддю для конкретного заданого тестового питання і порівняти цей варіант з тим, що видає на результат використаний алгоритм

3.2.4 Звіт результатів тестування

В таблиці нижче наведено результати виконання усіх загальних, функціональних та спеціальних тест-кейсів.

Таблиця 3.4 - Звіт тестування

Ідентифікатор тесту	Очікуваний результат	Фактичний результат	Статус тесту
GC-1	Бот починає роботу при виконанні команди /start	Бот починає роботу при виконанні команди /start	Пройдено
GC-2	При натисканні на кнопку із символом «/» або введенні цього ж символу вручну з’являється меню усіх команд, які може виконувати бот	При натисканні на кнопку із символом «/» або введенні цього ж символу вручну з’являється меню усіх команд, які може виконувати бот	Пройдено
GC-3	Повідомлення успішно надіслалося в мережу	Повідомлення успішно надіслалося в мережу	Пройдено
GC-4	Бот відреагував на надіслане повідомлення і надіслав необхідну відповідь	Бот відреагував на надіслане повідомлення і надіслав необхідну відповідь	Пройдено
GC-5	Бот зупиняється і стає неактивним після натиснення користувачем відповідної вбудованої команди одинаково в будь який момент виконання	Бот зупиняється і стає неактивним після натиснення користувачем відповідної вбудованої команди одинаково в будь який момент виконання	Пройдено

Продовження таблиці 3.4

Ідентифікатор тесту	Очікуваний результат	Фактичний результат	Статус тесту
GC-6	Історія повідомлень очистилась після виконання відповідної вбудованої команди що у робочому режимі боту, що після його зупинення	Історія повідомлень очистилась після виконання відповідної вбудованої команди що у робочому режимі боту, що після його зупинення	Пройдено
GC-7	При виході із діалогу з ботом і переході в інший діалог, бот залишається активним і вся інформація, введена користувачем зберігається	При виході із діалогу з ботом і переході в інший діалог, бот залишається активним і вся інформація, введена користувачем зберігається	Пройдено
FC-1	На повідомлення українською мовою з елементами вітальних слів, бот вивів шаблонне привітання у відповідь користувачу	На повідомлення українською мовою з елементами вітальних слів, бот вивів шаблонне привітання у відповідь користувачу	Пройдено
FC-2	Команда /add_link обирається із запропонованого списку і пропонує ввести ресурсне веб-посилання	Команда /add_link обирається із запропонованого списку і пропонує ввести ресурсне веб-посилання	Пройдено

Продовження таблиці 3.4

FC-3	Введене веб-посилання успішно записалося в базу даних	Введене веб-посилання успішно записалося в базу даних	Пройдено
FC-4	При виконанні парсингу у текстовий файл не виникло помилки про те, що програма не може опрацювати посилання	При виконанні парсингу у текстовий файл не виникло помилки про те, що програма не може опрацювати посилання	Пройдено
FC-5	При збереженні результуючого файлу локально, він відкривається не порожнім	При збереженні результуючого файлу локально, він відкривається не порожнім	Пройдено
FC-6	Команда /top_questions обирається із запропонованого списку і як результат виводить на екран список із 5 найчастіше задаваних питань	Команда /top_questions обирається із запропонованого списку і як результат виводить на екран список із 5 найчастіше задаваних питань	Пройдено
FC-7	При заданні одного і того ж питання у кількості 100 раз, воно має з’явитися в списку після виконання команди /top_questions	При заданні одного і того ж питання у кількості 100 раз, воно має з’явитися в списку після виконання команди /top_questions	Пройдено

Продовження таблиці 3.4

Ідентифікатор тесту	Очікуваний результат	Фактичний результат	Статус тесту
FC-8	Задається певне питання, яке ще не було у базі даних певну кількість раз і при запиті до бази даних отримане число співпадає з реальною кількістю разів, скільки його задавали	Задається певне питання, яке ще не було у базі даних певну кількість раз і при запиті до бази даних отримане число співпадає з реальною кількістю разів, скільки його задавали	Пройдено
FC-9	Команда /rate обирається із запропонованого списку і пропонує користувачу кнопки з градацією оцінок (від 1 до 5)	Команда /rate обирається із запропонованого списку і пропонує користувачу кнопки з градацією оцінок (від 1 до 5)	Пройдено
FC-10	При натисканні кнопки з бажаною оцінкою після виконання /rate, оцінка зчитується серверною частиною і далі аналізується	При натисканні кнопки з бажаною оцінкою після виконання /rate, оцінка зчитується серверною частиною і далі аналізується	Пройдено

Продовження таблиці 3.4

Ідентифікатор тесту	Очікуваний результат	Фактичний результат	Статус тесту
FC-11	Після того, як користувач ввів питання, провівся аналіз і бот видав найбільш імовірну відповідь	Після того, як користувач ввів питання, провівся аналіз і бот видав найбільш імовірну відповідь	Пройдено
FC-12	Пошуком по документах, знаходимо серед присутніх в базі той, з якого була взята відповідь	Пошуком по документах, знаходимо серед присутніх в базі той, з якого була взята відповідь	Пройдено
FC-13	Якщо база даних залишалась без змін, при введені одного і того ж питання багаторазово, відповідь залишалася незмінною	Якщо база даних залишалась без змін, при введені одного і того ж питання багаторазово, відповідь залишалася незмінною	Пройдено
FC-14	При введенні користувачем питання, воно успішно записалося в базу даних, тому при запиті до бази, зможемо його знайти	При введенні користувачем питання, воно успішно записалося в базу даних, тому при запиті до бази, зможемо його знайти	Пройдено

Продовження таблиці 3.4

Ідентифікатор тесту	Очікуваний результат	Фактичний результат	Статус тесту
FC-15	Після виконання команди /add_link, введене користувачем записання записалося до бази дааних і знаходиться при прямому запиті до бази	Після виконання команди /add_link, введене користувачем записання записалося до бази дааних і знаходиться при прямому запиті до бази	Пройдено
FC-16	Після парсингу результуючий файл зберегти локально і порівняти змістовність файлу з інформацією по відповідному веб-посиланню	Після парсингу результуючий файл зберегти локально і порівняти змістовність файлу з інформацією по відповідному веб-посиланню	Пройдено
FC-17	Отриманий в результаті парсингу файл знаходиться при прямому зверненні до бази даних	Отриманий в результаті парсингу файл знаходиться при прямому зверненні до бази даних	Пройдено
FC-18	При введенні питання, яке повторює текст щойнозаписаного файлу, відповідь видається саме з цього файлу	При введенні питання, яке повторює текст щойнозаписаного файлу, відповідь видається саме з цього файлу	Пройдено

Продовження таблиці 3.4

Ідентифікатор тесту	Очікуваний результат	Фактичний результат	Статус тесту
SC-1	Тестовий приклад тексу розбився на окремі речення	Тестовий приклад тексу розбився на окремі речення	Пройдено
SC-2	Текстовий приклад тексту розбився на окремі токени у вигляді слів	Текстовий приклад тексту розбився на окремі токени у вигляді слів	Пройдено
SC-3	Після процесу лематизації україномовний текст, розбитий на окремі токени привівся до загальної форми	Після процесу лематизації україномовний текст, розбитий на окремі токени привівся до загальної форми	Пройдено
SC-4	Тестовий приклад тексту після опрацювання до вигляду окремих лем очищається від непотрібних лем, які визначені, як стоп-слова	Тестовий приклад тексту після опрацювання до вигляду окремих лем очищається від непотрібних лем, які визначені, як стоп-слова	Пройдено
SC-5	Застосовуючи регулярні вирази, текст очистився від пунктуації та інших символів	Застосовуючи регулярні вирази, текст очистився від пунктуації та інших символів	Пройдено

Продовження таблиці 3.4

Ідентифікатор тесту	Очікуваний результат	Фактичний результат	Статус тесту
SC-6	Після підрахунків частотності у межах двох файлів за принципом, який застосовує метод TF-IDF, вони співпали з результатом, який видає бібліотечний метод TF-IDF	Після підрахунків частотності у межах двох файлів за принципом, який застосовує метод TF-IDF, вони співпали з результатом, який видає бібліотечний метод TF-IDF	Пройдено
SC-7	Обрана відповідь на конкретне тестове запитання співпала з відповіддю, яку надав алгоритм	Обрана відповідь на конкретне тестове запитання співпала з відповіддю, яку надав алгоритм	Пройдено

3.3 Опис контрольного прикладу

У якості контрольного прикладу розглянемо тест-кейс FC-9, який описує процес виставлення оцінок знайденим відповідям самими користувачами. Для даної задачі це є найбільш розумний і достовірний метод тестування роботи системи в цілому з точки зору кінцевого користувача.

В ході тестування було зібрано біля 50 оцінок від різних користувачів тестової групи. Оцінки проводилися по 5-бальній шкалі з таким розподілом :

- 1 – відповідь зовсім не співпадає з темою запиту;
- 2 – відповідь наближено співпадає з темою запиту;
- 3 – відповідь частково співпадає з темою запиту;

– 4 – відповідь майже співпадає з темою запиту;
– 5 – відповідь повністю співпадає з темою запиту.

3.4. Висновки по розділу

У ході процесу тестування було розроблено три різновиди тест-кейсів – загальні, функціональні та спеціальні. Згідно цього плану було проведено тестування розробленого програмного продукту. Опираючись на результати тестування, які були представлені у вигляді таблиці, можемо зробити висновки, що програмний застосунок працює коректно і його якість можна назвати задовільною.

4 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Розгортання програмного забезпечення

Розроблене програмне забезпечення може бути розгорнуте на операційній системі Windows. Для цього на пристрої має бути встановлений Python версії 3.5 або вище. На базі Python також має бути встановлено ряд бібліотек для роботи з наявним функціоналом:

- бібліотеки для роботи з текстами та здійснення аналізу із застосуванням NLP (nltk, numpy, pymorphy2, TextPreprocessor, sklearn.feature_extraction.text);
- засоби роботи з Telegram API (telebot);
- бібліотеки для веб-скрапінгу і парсингу (requests, contextlib, requests.exceptions, bs4).

При встановленні pymorphy2 і TextPreprocessor можуть виникнути труднощі з версійністю, тому рекомендовано використовувати останні версії pip ти setuptools.

Також для зручної роботи з великою кількістю даних і тестування рекомендується встановити СУБД PostgreSQL.

4.2 Робота з програмним забезпеченням

Детальний опис роботи з розробленим програмним забезпеченням та наглядна інструкція наведені у розділі «Керівництво користувача».

ВИСНОВКИ

В рамках даного дипломного проекту було розроблено програмне забезпечення підтримки абітурієнтів. Розробка включала в себе проведення аналізу предметної області, а саме аналіз різноманітних діалогових систем і чат-ботів. Також було зібрано датасет, серед даних якого проводиться пошук згідно використаних алгоритмів. Під час збору інформації було проведено глибокий аналіз сучасних засобів роботи з абітурієнтами та їх інформування в рамках НТУУ «КПІ».

Робота містить 4 розділи, які повною мірою описують всі етапи виконання дипломного проекту.

В першому розділі описано аналіз вимог до програмного забезпечення, як функціональних, так і нефункціональних. Наявний огляд предметної області, відомих ІТ-продуктів, терміни та скорочення, які грають вагому роль у розумінні тематики розробки.

В другому розділі описується архітектура та конструювання власне самої програми. Відображено та розтлумачено усі складові результуючого програмного продукту та розказано про засоби для забезпечення цілісності та безпеки даних користувачів в рамках розробки.

Під час роботи над третім розділом було розроблено тест-кейси та на їх базі проведено повне тестування отриманого програмного забезпечення. Наглядно показано всі кроки тестування, результати та розглянуто контрольний приклад. За допомогою цього, ми змогли підтвердити якість та роботоздатність розробки.

В четвертому розділі демонструється, як потрібно розгортати програмне забезпечення, що включає в себе процес запуску програми та де можна знайти керівництво для успішної взаємодії користувача з інтерфейсом.

Також було змодельовано UML-діаграму класів, UseCase діаграму і схему структури бази даних, які можна знайти у додатках.

ПЕРЕЛІК ПОСИЛАНЬ

1) Тематическое моделирование текстовых коллекций в диалоговых системах [Електронний ресурс]: (Стаття) // Московский государственный университет имени М.В. Ломоносова. – 2018. – Режим доступу до ресурсу: http://www.machinelearning.ru/wiki/images/3/32/Kruglikov18bsc.pdf.

2) Jurafsky D. Question Answering / D. Jurafsky, J. Martin // Speech and Language Processing / D. Jurafsky, J. Martin., 2019.

3) Основы Natural Language Processing для текста [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://habr.com/ru/company/Voximplant/blog/446738/.

4) Manning C. Introduction to Information Retrieval / C. Manning, P. Raghavan, H. Schütze., 2008.

5) Swalin A. Building a Question-Answering System from Scratch [Електронний ресурс] / Alvira Swalin // Part 1. – 2018. – Режим доступу до ресурсу: https://towardsdatascience.com/building-a-question-answering-system-part-1-9388aadff507.

6) Understanding TF-IDF (Term Frequency-Inverse Document Frequency) in python [Електронний ресурс] – Режим доступу до ресурсу: https://medium.com/@azunan3/understanding-tf-idf-term-frequency-inverse-document-frequency-in-python-373070acb895.

7) Korobov M. Morphological Analyzer and Generator for Russian and Ukrainian Languages / Mikhail Korobov. // ScrapingHub, Inc..

8) Vembunarayanan J. Tf-Idf and Cosine similarity [Електронний ресурс] / Jana Vembunarayanan. – 2013. – Режим доступу до ресурсу: https://janav.wordpress.com/2013/10/27/tf-idf-and-cosine-similarity/.

9) TF-IDF [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/TF-IDF.

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____О.А. Павлов

“___”_____2020 р.

ДІАЛОГОВА СИСТЕМА ІНФОРМУВАННЯ АБІТУРІЄНТІВ

Технічне завдання

КПІ.ІП-6302.045430.03.91

“ПОГОДЖЕНО”

Керівник проекту:

_____О.К. Очеретяний

Нормоконтроль:

_____ К.І. Ліщук

Виконавець:

_____В.С. Бондар

Київ – 2020 року

ЗМІСТ

НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ 4

2 ПІДСТАВА ДЛЯ РОЗРОБКИ 5

3 ПРИЗНАЧЕННЯ РОЗРОБКИ 6

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 7

4.1 Вимоги до функціональних характеристик 7

4.2 Вимоги до надійності 6

4.3 Умови експлуатації 6

4.4 Вимоги до складу і параметрів технічних засобів 7

4.5 Вимоги до інформаційної та програмної сумісності 7

4.6. Вимоги до маркування та пакування 7

4.7 Вимоги до транспортування та зберігання 7

4.8 Спеціальні вимоги 7

5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ 8

6 СТАДІЇ І ЕТАПИ РОЗРОБКИ 9

7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ 10

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Діалогова система інформування абітурієнтів

Галузь застосування: Наведене технічне завдання поширюється на розробку програмного забезпечення «Діалогова система інформування абітурієнтів» [КПІ.ІП-6302.045430.03.91], котра використовується для інформування абітурієнтів КПІ по заданій тематиці у вигляді Telegram чат-боту, пошуку і видачі релевантних відповідей на поставлені користувачем питання та збір статистики по найбільш часто задаваним питанням та призначена для абітурієнтів, які зацікавлені у вступі до КПІ.

2 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки «Діалогова система інформування абітурієнтів» є завдання на дипломне проектування, затверджене кафедрою автоматизованих систем обробки інформації та управління Національного технічного університету України «Київський політехнічний інститут ім. Ігоря Сікорського» (НТУУ «КПІ»).

3 ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для абітурієнтів та майбутніх студентів КПІ, які прагнуть отримати необхідну їм інформацію щодо вступу та навчанню в університеті тощо.

Метою розробки є створення єдиного програмного забезпечення, яке дозволить отримувати необхідну для користувача інформацію щодо вступу та навчанню у КПІ за допомогою методів семантичного аналізу тексту та NLP.

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

4.1.1 Програмне забезпечення повинно забезпечувати виконання наступних основних функції:

4.1.1.1 Для користувача:

– взаємодія з інтерфейсом Telegram боту за допомогою елементарних команд;

– семантичний аналіз дата сету по заданому питанню;

– виведення найбільш підходящої відповіді серед наявних даних;

– виведення п’яти найбільш часто задаваних питань

– Додавання користувачем веб-посилання на ресурс

4.1.2. Розробку виконати на платформі Windows

4.2. Вимоги до надійності

4.2.1. Передбачити контроль введення інформації.

4.2.2. Передбачити захист від некоректних дій користувача.

4.2.3. Забезпечити цілісність інформації в базі даних.

4.3. Умови експлуатації

Для користування системою необхідно підключення до мережі Інтернет.

4.3.1. Умови експлуатації згідно СанПін 2.2.2.542 – 96.

4.3.2. Обслуговування

Даний продукт потребує обслуговування лише з точки зору поповнення і оновлення існуючого дата сету.

4.3.3. Обслуговуючий персонал

Для обслуговування достатньо одного адміністратора або додатковогопрограмного продукту (скрипта), який буде виконувати обслуговування автоматично. 4.4. Вимоги до складу і параметрів технічних засобів

4.4.1. Програмне забезпечення повинно функціонувати на IBM-сумісних персональних комп‘ютерах.

4.4.2. Мінімальна конфігурація технічних засобів:

4.4.2.1. Тип процесору .....86_64.

4.4.2.2. Об‘єм ОЗП ..... 1024Мб.

4.4.2.3. Достатній об’єм жорсткого диску ..... 1 Гб.

4.5. Вимоги до інформаційної та програмної сумісності

4.5.1. Програмне забезпечення повинно працювати під управлінням операційних систем сімейства WIN32 (Windows'XP, Windows NT і т.д.).

4.5.2. Вхідні дані повинні бути представлені у форматі текстового повідомлення у діалоговому вікні створеного Telegram чат-боту.

4.5.3. Результати повинні бути представлені у форматі отриманого текстового повідомлення зі сторони Telegram боту.

4.5.4. Програмне забезпечення повинно бути представленим за допомогою обгортки Telegram Bot API.

4.6. Вимоги до маркування та пакування

Вимоги до маркування та пакування не пред‘являються.

4.7. Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання не пред'являються.

4.8. Спеціальні вимоги

Згенерувати установчу версію програмного забезпечення.

5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

5.1. Програмні модулі, котрі розробляються, повинні бути задокументовані, тобто тексти програм повинні містити всі необхідні коментарі.

5.2. Програмне забезпечення повинно мати довідникову систему

5.3. У склад супроводжувальної документації повинні входити наступні документи:

5.3.1. Пояснювальна записка не менше ніж на 100 аркушах формату А4 (без додатків 5.3.2 - 5.3.6).

5.3.2. Технічне завдання.

5.3.3. Керівництво користувача.

5.3.4. Програма та методика тестування

5.4. Графічна частина повинна бути виконана на листах формату А3, котрі включаються у якості додатків до пояснювальної записки:

5.4.1. Схема бази даних.

5.4.2. Схема структурна класів програмного забезпечення.

5.4.3. Схема структурна варіантів використання.

6 СТАДІЇ І ЕТАПИ РОЗРОБКИ

№	Назва етапу	Строк	Звітність
1.	Вивчення літератури за тематикою проекту	21.02.2020	
2.	Розробка технічного завдання	03.03.2020	Технічне завдання
3.	Аналіз вимог та уточнення специфікацій	19.03.2020	Специфікації програмного забезпечення
4.	Проектування структури програмного забезпечення, проектування компонентів	30.03.2020	Схема структурна програмного забезпечення та специфікація компонентів (діаграма класів, схема алгоритму …)
5.	Програмна реалізація програмного забезпечення	05.04.2020	Тексти програмного забезпечення
6.	Тестування програмного забезпечення	10.04.2020	Тести, результати тестування
7.	Розробка матеріалів текстової частини проекту	14.04.2020	Пояснювальна записка.
8.	Розробка матеріалів графічної частини проекту	20.04.2020	Графічний матеріал проекту
9.	Оформлення технічної документації проекту	29.04.2020	Технічна документація

7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

7.1 Види випробувань

Тестування розробленого програмного продукту виконується відповідно до підрозділу «Випробування програмного продукту».

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____Олександр ПАВЛОВ

“___”_____2020 р.

ДІАЛОГОВА СИСТЕМА ІНФОРМУВАННЯ АБІТУРІЄНТІВ

Опис програми

КПІ.ІП-6302.045430.04.13

“ПОГОДЖЕНО”

Керівник проєкту:

_____О.К. Очеретяний

Нормоконтроль:

_____ К.І. Ліщук

Виконавець:

_____В.С. Бондар

Київ – 2020 року

Тексти програмного коду

Діалогова система інформування абітурієнтів

(TelegramInteraction)

```
from random import random
import telebot

bot = telebot.TeleBot('BOT_ID')
GREETING_INPUTS = ("привіт", "привет", "хай", "hi",
"hello","добрий день")
GREETING_RESPONSES = ["Привіт!", "Хай :)", "Привітики!",
"Здоров був", "Раді тебе бачити!", "Привіт, друже)"]


@bot.message_handler(commands=['start'])
def start_message(message):
    bot.send_message(message.chat.id, 'Привіт! Я тут, що
допомогти тобі дізнатися щось про вступ до КПІ. '
                                      'Задай мені будь-яке
питання, яке тебе цікавить з цього приводу.')


@bot.message_handler(commands=['add_link'])
def add_link_message(message):
    bot.send_message(message.chat.id, 'Якщо ти не отримав
відповідь на питання тут, '
                                      'але знашов її на
просторах інтернету, '
                                      'то можеш надіслати веб-
посилання на той ресурс в наступному повідомлені.'
                                      'В іншому випадку напиши
"Ні"')
    if message.text.lower() == 'ні':
        bot.send_message(message.chat.id,'Повертайся, якщо
знайдеш щось цікаве!')
    else:
        bot.send_message(message.chat.id, 'Дякуємо, тепер
інформація по посиланню в базі даних цього боту!')
```

```
@bot.message_handler(commands=['top_questions'])
def top_questions_message(message):
    bot.send_message(message.chat.id,'Що   пріоритет? \n'
                                     'Якого числа починається вступнв кампанія? \n'
                                     'Чи існує вступ поза конкурсом? \n'
                                     'Який мінімальний прохідний бал у КПІ? \n'
                                     'Що таке КПІ?')


@bot.message_handler(commands=['rate'])
def top_questions_message(message):
    bot.send_message(message.chat.id, 'Тут ти можеш оцінити, наскільки знайдені ботом відповіді '
                                      'співпадали з твоїми очікуваннями: ')
    keyboard1 = telebot.types.ReplyKeyboardMarkup(True, True)
    keyboard1.row('1', '2', '3', '4', '5')
    bot.send_message(message.chat.id, 'Дякуємо, що ти з нами!')


@bot.message_handler(content_types=['text'])
def send_text(message):
    if message.text.lower() == greeting(message):
        bot.send_message(message.chat.id, greeting(message))
```

```
    if message.text.lower() == 'що таке пріоритет?':
        bot.send_message(message.chat.id, 'Пріоритет — показник, виражений числами від 1 до 5, '
                                          'який вступник особисто присвоює своїм заявам, '
                                          'де 1 є показником найбільшої пріоритетності заяви, '
                                          'тобто на першому місці та спеціальність, '
                                          'на яку ти хочеш вступити найбільше. '
                                          'Пріоритетність визначається під час вступу на навчання на денну та заочну '
                                          'форми навчання на місця державного замовлення на основі '
                                          'повної загальної середньої освіти.')


def greeting(sentence):
    """If user's input is a greeting, return a greeting response"""
    for word in sentence.split():
        if word.lower() in GREETING_INPUTS:
            return random.choice(GREETING_RESPONSES)
```

Тексти програмного коду

Діалогова система інформування абітурієнтів

(DBOperations)

```
import pytz

import requests

import datetime

import newspaper


class LinkItem (models.Model):

 link = models.CharField(max_length=100, null=True,
blank=True)

 fullname = models.CharField(max_length=200, null=True,
blank=True)

 logo = models.CharField(max_length=400, null=True,
blank=True)

 bio = models.CharField(max_length=1000, null=True,
blank=True)


class IncomingQuestion (models.Model):

 message = models.CharField(max_length=100, null=True,
blank=True)

 useerID  = models.CharField(max_length=200, null=True,
blank=True)

 username = models.CharField(max_length=200, null=True,
blank=True)

 score = models.CharField(max_length=400, null=True,
blank=True)

 question = models.CharField(max_length=1000, null=True,
blank=True)
```

```
class UserScore (models.Model):
 userID = models.CharField(max_length=100, null=True,
blank=True)
 username = models.CharField(max_length=200, null=True,
blank=True)
 questionID = models.CharField(max_length=400, null=True,
blank=True)
 score = models.CharField(max_length=1000, null=True,
blank=True)


class DocumentItem (models.Model):
 name = models.CharField(max_length=100, null=True,
blank=True)
 path = models.CharField(max_length=200, null=True,
blank=True)
 link = models.CharField(max_length=400, null=True,
blank=True)
 document = models.CharField(max_length=1000, null=True,
blank=True)
```

Тексти програмного коду

Діалогова система інформування абітурієнтів

(DataProcessing)

```
import nltk as nltk
from pymorphy2 import MorphAnalyzer
import numpy
from TextPreprocessor import sent_tokenizer_ru,
word_tokenizer, stop_words_ru, sent_tokenizer_ua,
stop_words_ua


#  Reading in the corpus
with open('scraped_data.txt', 'r', encoding='utf8',
errors='ignore') as fin:
    raw = fin.read().lower()


def summarize(text, language, n=5):
    morph = MorphAnalyzer()
    sent_tokenizer = sent_tokenizer_ru
    stop_words = stop_words_ru

    if language == 'ukrainian':
        morph = MorphAnalyzer(lang='uk')
        sent_tokenizer = sent_tokenizer_ua
        stop_words = stop_words_ua

    d = create_dictionary(text, morph, stop_words)
    m = create_matrix(text, sent_tokenizer, morph, d)
    r = compute_ranks(m, n)
    sentences = sent_tokenizer(text)
    rank_sort = sorted(((i, r[i], s) for i, s in enumerate(
        sentences)), key=lambda x: r[x[0]], reverse=True)
    top_n =  sorted(rank_sort[:n])
    return ' '.join(x[2] for x in top_n)

#Tokenisation
# converts to list of sentences
word_tokens = nltk.word_tokenize(raw)# converts to list of
words
```

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____Олександр ПАВЛОВ

“___”_____2020 р.

ДІАЛОГОВА СИСТЕМА ІНФОРМУВАННЯ АБІТУРІЄНТІВ

Програма та методика тестування

КПІ.ІП-6302.045430.05.51

“ПОГОДЖЕНО”

Керівник проєкту:

_____О.К. Очеретяний

Нормоконтроль:

_____ К.І. Ліщук

Виконавець:

_____В.С.Бондар

Київ – 2020 року

ЗМІСТ

1 ОБ'ЄКТ ВИПРОБУВАНЬ 3

2 МЕТА ТЕСТУВАННЯ 4

3 МЕТОДИ ТЕСТУВАННЯ 5

4 ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ..... 6

1 ОБ’ЄКТ ВИПРОБУВАНЬ

Діалогова система інформування абітурієнтів, яка являє собою Telegram бот, створений за допомогою мови програмування Python з використанням ряду відповідних бібліотек для різних задач, таких як numpy, nltk, pymorphy2 та інших.

2 МЕТА ТЕСТУВАННЯ

У процесі тестування має бути перевірено наступне:

– функціональна працездатність елементів взаємодії з ботом;

– коректність виконання команд;

– надання коректної відповіді з максимально можливою відповідністю очікуваному результату;

– коректне виконання парсингу web-сорінки у текстовий файл;

– зручність роботи з сервісом;

– відповідність програмного комплексу вимогам Технічного завдання.

3 МЕТОДИ ТЕСТУВАННЯ

Тестування виконується методом Gray Box Testing. Перевіряється як код, так і безпосередньо програмний продукт на відповідність функціональним вимогам. Тестування методом чорного ящика передбачає тестування з позиції користувача. Це означає, що в процесі тестування доступ до внутрішньої структури програми і роботи алгоритмів не передбачається.

Використовуються наступні методи:

– дослідницьке тестування;
– функціональне тестування;
– тестування інтерфейсу користувача.

4 ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Тестування проводиться за допомогою засобів бібліотеки Unittest.

Працездатність застосування перевіряється шляхом:

- динамічного ручного тестування – введенням граничних та недопустимих значень у діалозі з ботом ;
- динамічного ручного тестування на відповідність функціональним вимогам;
- статичного тестування коду;
- тестування коректності обчислень за допомогою підготовних наборів даних;
- тестування стабільності роботи при різних умовах;
- тестування зручності використання;
- тестування інтерфейсу.

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____Олександр ПАВЛОВ

“___”_____2020 р.

ДІАЛОГОВА СИСТЕМА ІНФОРМУВАННЯ АБІТУРІЄНТІВ

Керівництво користувача

КПІ.ІП-6302.045430.06 .34

“ПОГОДЖЕНО”

Керівник проєкту:

_____О.К. Очеретяний

Нормоконтроль:

_____ К.І. Ліщук

Виконавець:

_____В.С. Бондар

Київ – 2020 року

Для того, щоб узагалі почати роботу із програмою, потрібно перш за все відкрити Telegram. Це може бути десктоп, веб або мобільна версія – на роботу боту це ніяк не впливає.

Наступним кроком буде знайти в мережі Telegram створеного бота за юзернеймом @AbitSupportBot. На зображенні нижче можна побачити знайденого бота та інормацію про нього:

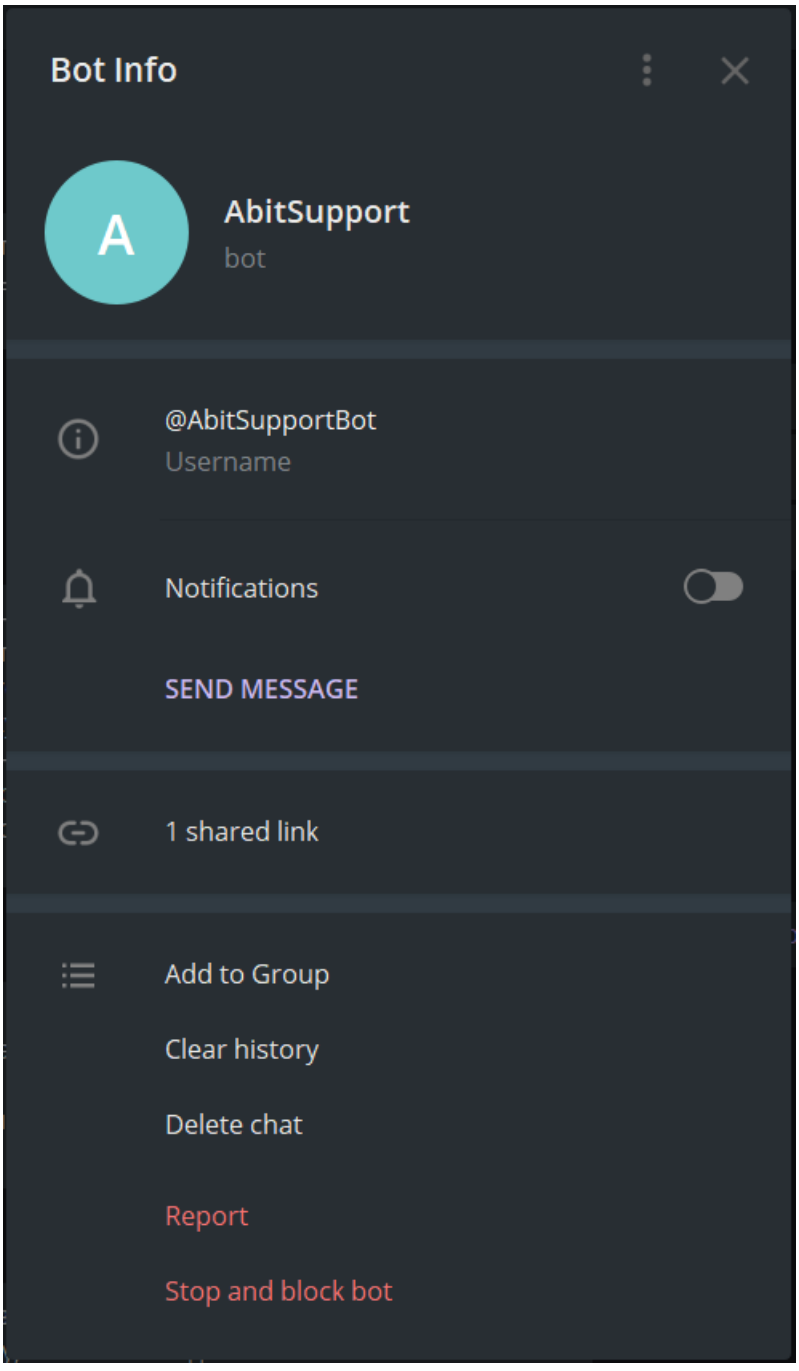


Рисунок 1 – Інформаційне вікно боту

Початок роботи безпосередньо з ботом відбувається через стандартну для Telegram команду */start*. Результати її виконання і запуск роботи боту зображено на наступному малюнку:

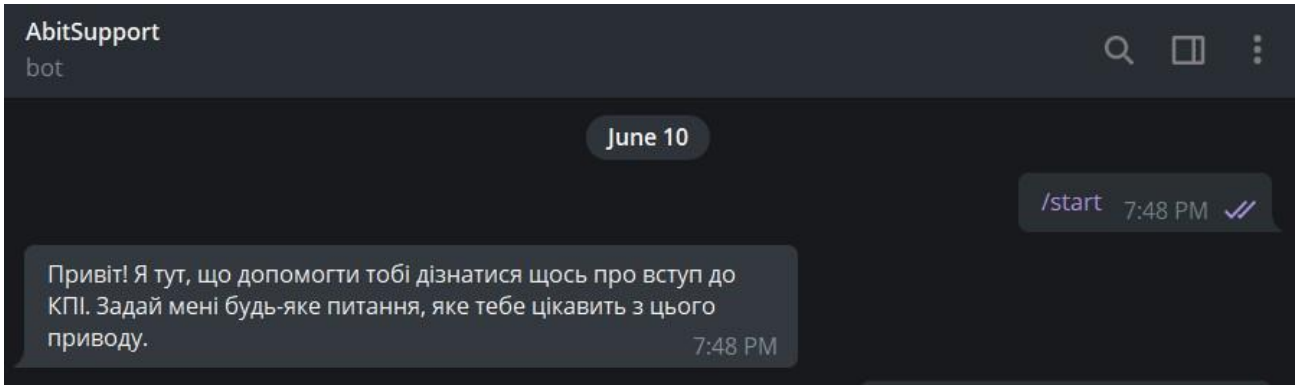


Рисунок 2 – Запуск боту

Далі у користувача є вибір - або скористатися однією із вбудованих команд, або просто задати боту будь-яке питання у вільній формі. Так як основне призначення розробки – це відповідь на питання, то очікуваніше, що користувач задасть цікавляче його питання в першу чергу. Рисунок 3 демонструє, якою простою є взаємодія з ботом, тому що майже весь функціонал і всі обрахунки сховані під фасадом:

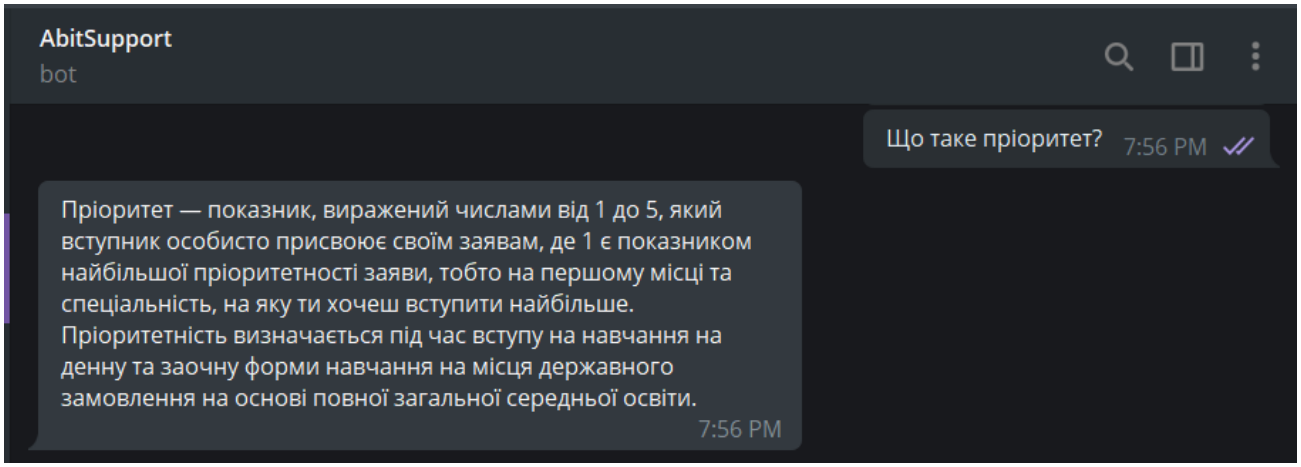


Рисунок 3 – Отримання результатів на задане питання

Весь інший функціонал, який розроблявся в рамках цієї дипломної роботи реалізований за допомогою вбудованих команд.

Перша команда, яку варто розглянути – це */top_questions*. Команда розрахована на те, що користувач із бажанням почати працювати з ботом, може не знати одразу, які питання задати. Результатом виконання команди є виведення у повідомленні списку із 5 питань, які задаються користувачами найчастіше. На Рисунку 4 зображено результати виконання команди */top_questions:*

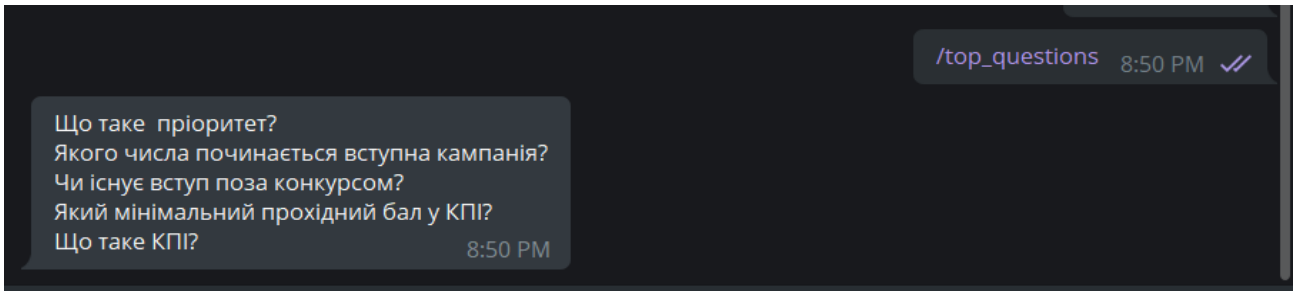


Рисунок 4 – Виведення на екран найпопулярніших запитань

Наступний функціонал надає змогу користувачу поповнювати базу даних текстів, серед яких проводиться аналіз. Наприклад, якщо користувач вів запитання, і серед наявних в базі ресурсів програма не змогла знайти підходящої відповіді. Тоді якщо при цьому користувач знайшов відповідь в Інтернеті, він може занести до бази посилання, яке автоматично пропарсить дані з веб-сайту в текстовий файл і збереже його до бази. Тобто наступного разу, коли хтось зробить такий самий запит, відповідь уже віднайдеться. Це вирішує проблему дефіциту даних і дозволяє програмі працювати більш автономно. Як цей функціонал виглядає для користувача демонструє наступний рисунок:

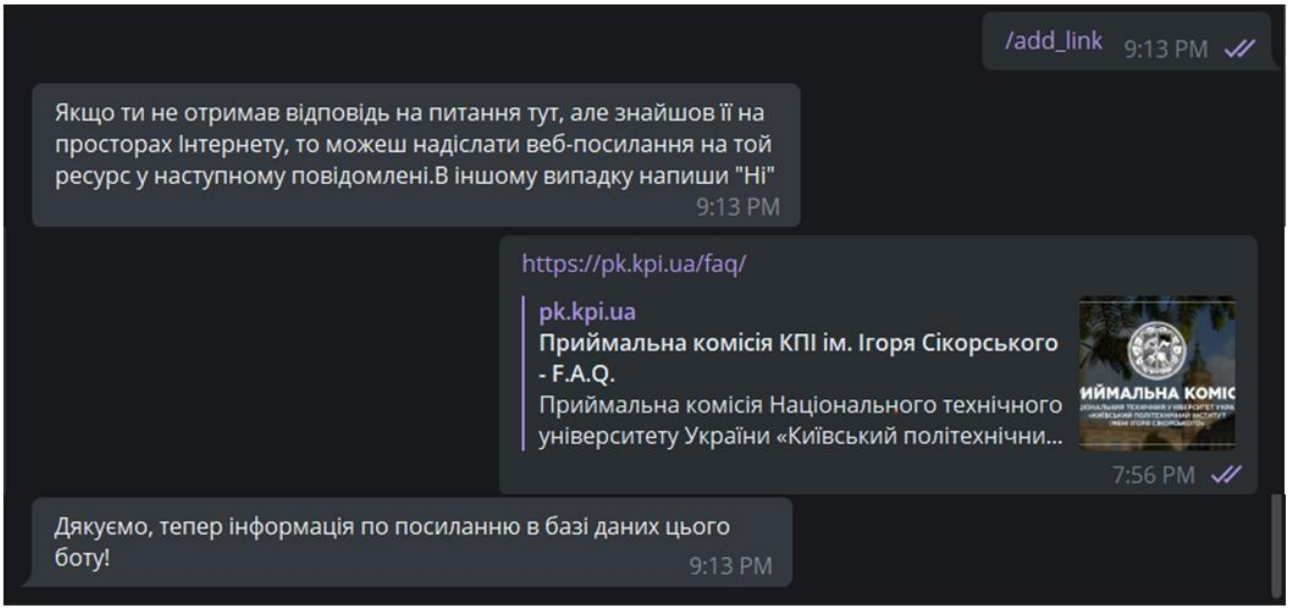


Рисунок 5 – Додавання веб-посилання на ресурс інформації

Тестування релевантності знайдених відповідей складно зробити програмним шляхом. Тому був створений функціонал, який дозволяє надати оцінку точності, збираючи відгуки напряму від користувача. Поставити оцінку відповідності отриманого результату до бажаного можна за допомогою команди */rate*. З допомогою спеціальної панелі з оцінками користувач досить просто може поставити відповідну оцінку :

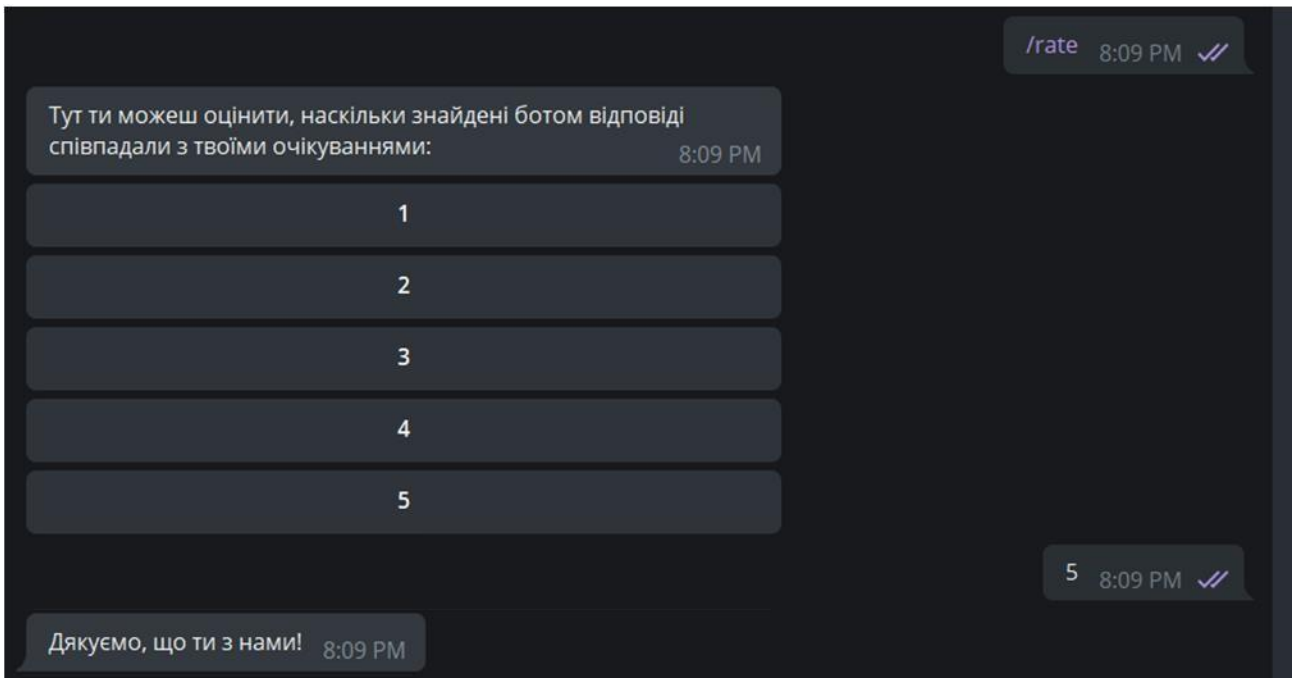


Рисунок 6 – Проведення користувачем оцінки точності відповідей

Увесь інший функціонал, який наявний у боті є вбудованим у сам месенджер Telegram і не має відношення до даної розробки.

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____Олександр ПАВЛОВ

“___”_____2020 р.

ДІАЛОГОВА СИСТЕМА ІНФОРМУВАННЯ АБІТУРІЄНТІВ

Графічний матеріал

КПІ.ІП-6302.045430.07.99.СС

“ПОГОДЖЕНО”

Керівник проєкту:

_____О.К. Очеретяний

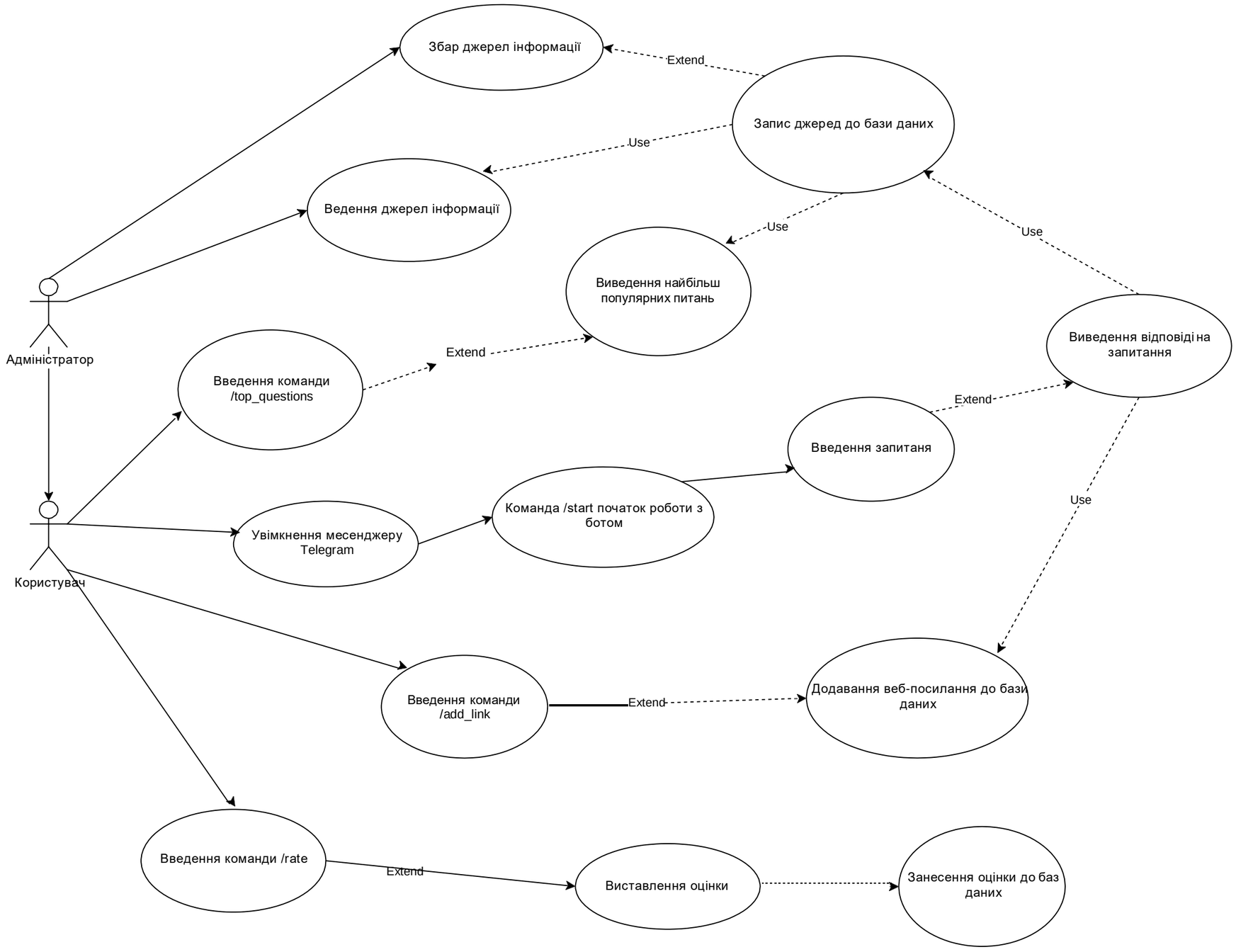
Нормоконтроль:

_____ К.І. Ліщук

Виконавець:

_____В.С.Бондар

Київ – 2012 року

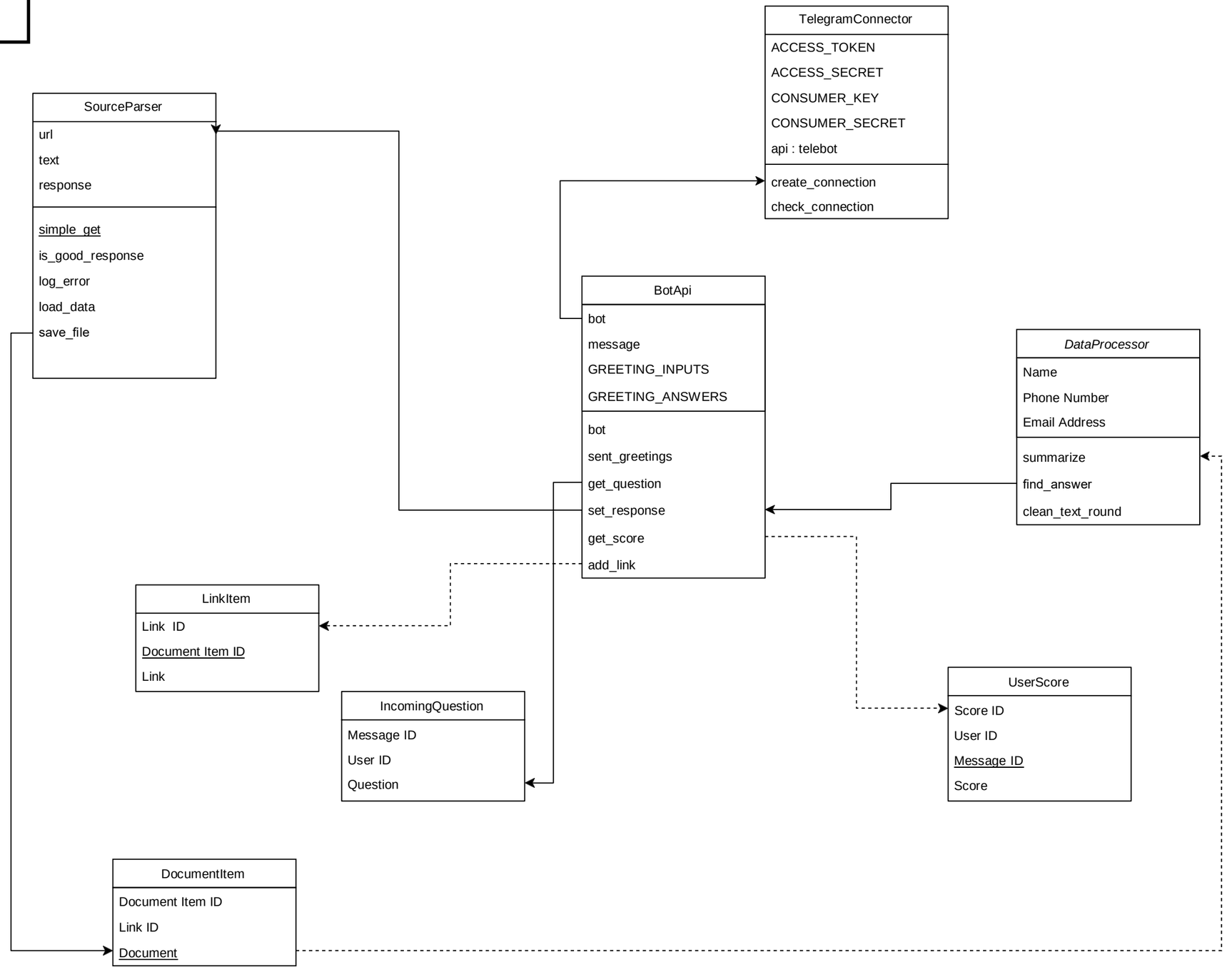


					КПІ.ІП-6302.045430.07.99.СС	
Зм.	Арк.	№ докум.	Підп.	Дата	Схема структурна варіантів використань	Лит. / Арк. / Аркушів: 1
Розроб.		Бондар В.С.				Аркуш / Аркушів
Перев.		Очеретяний О.К.			Діалогова система інформування абітурієнтів	КПІ ім.Ігоря Сікорського Кафедра АСОІУ гр. ІП-63
Т. Кон.						
Н. Кон.		Ліщук К.І.				
Затв.		Очеретяний О.К.				

КПІ.ІП-6302.045430.08.99.СБД

					КПІ.ІП-6302.045430.08.99.СБД	
Зм.	Арк.	№ докум.	Підп.	Дата		
Розроб.		Бондар В.С.			Схема бази даних	Лит. / Арк. / Аркушів: 1
Перев.		Очеретяний О.К.				
Т. Кон.						Аркуш / Аркушів
					Діалогова система інформування абітурієнтів	КПІ ім.Ігоря Сікорського Кафедра АСОІУ гр. ІП-63
Н. Кон.		Ліщук К.І.				
Затв.		Очеретяний О.К.				

КПІ.ІП-6302.045430.09.99.СС



					КПІ.ІП-6302.045430.09.99.СС			
						Лит.	Арк.	Аркушів
Зм.	Арк.	№ докум.	Підп.	Дата	Схема структурна класів програмного забезпечення			1
Розроб.		Бондар В.С.				Аркуш	Аркушів	
Перев.		Очеретяний О.К.			Діалогова система інформування абітурієнтів	КПІ ім.Ігоря Сікорського Кафедра АСОІУ гр. ІП-63		
Т. Кон.								
Н. Кон.		Ліщук К.І.						
Затв.		Очеретяний О.К.						